

**Daniele Miorandi, Louay Bassbouss, Heiko Pfeffer,
Paolo Dini, Gabor Horvath, [Silvia Elaluf-Calderwood](#),
Daniel Schreckling, Lidia Yamamoto**

BIONETICS D.3.2.7: autonomic services within the BIONETS SerWorks architecture

Project report

Original citation:

Miorandi, D., Bassbour, Louay, Pfeffer, H., Dini, Paolo, Horvarth, G., Elaluf-Calderwood, Silvia, Schreckling, D. and Yamamoto, L. (2009) *BIONETICS D.3.2.7: autonomic services within the BIONETS SerWorks architecture*. Public Deliverables, Lahti, Janne and Huusko, Jyrki (eds.) BIONETS/LSE/D3.2.7. BIONETS, London, UK.

Originally available from [BIONETS](#)

This version available at: <http://eprints.lse.ac.uk/54770/>

Available in LSE Research Online: March 2014

© 2009 The Authors

LSE has developed LSE Research Online so that users may access research output of the School. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Users may download and/or print one copy of any article(s) in LSE Research Online to facilitate their private study or for non-commercial research. You may not engage in further distribution of the material or use it for any profit-making activities or any commercial gain. You may freely distribute the URL (<http://eprints.lse.ac.uk>) of the LSE Research Online website.



BIONETS

WP 3.2 – AUTONOMIC SERVICE LIFE-CYCLE AND SERVICE ECOSYSTEMS

Autonomic Services within the BIONETS SerWorks Architecture

Deliverable 3.2.7

Reference:	BIONETS/VTT/WP3.2/v1.1
Category:	Deliverable
Editor:	Janne Lahti (VTT), Jyrki Huusko (VTT)
Co-authors:	Daniele Miorandi (CN), Louay Bassbouss (TUB), Heiko Pfeffer (TUB), Paolo Dini (LSE), Gabor Horvath (LSE), Silvia Elaluf-Calderwood (LSE), Daniel Schreckling (UNIPASSAU), Lidia Yamamoto (UBasel).
Verification:	Ludovic Henrio (INRIA), László Dóra (BUTE)
Date:	December 29, 2009
Status:	Final
Availability:	Public

Executive Summary

In this deliverable we summarize the research work and concepts of BIONETS subproject 3 (SP3) for topics related to BIONETS autonomic service life-cycle and service ecosystem, and provide final outcome of SP3 with regard to a novel life-cycle support that enables services to evolve over time to remain responsive to environmental changes. The deliverable extends the BIONETS SerWorks Service Framework defined in deliverables [BIONETS_D113/313] and [BIONETS_D114/314] by illustrating some of the main innovations behind the SerWorks solutions and its service Framework.

Since the work in BIONETS SP3 has been distributed in several different deliverables during the project lifetime and provided quite a amount of good solutions for the stated problems, this deliverable targets to provide a comprehensive overview and summary of the BIONETS service framework solutions and main features.

In the first part of the deliverable we concentrate on the SP3 solutions for service framework and service life-cycle. These solutions includes for example the MAPE (monitor, analysis, plan and execute) control loop definition for BIONETS services taking into consideration the fitness evaluation and embryogenic solutions of BIONETS. In addition we illustrate the socio-economic metaphors and evolving business models for BIONETS services and discuss about the possibilities of BIONETS self-contained and coupled business/economic models. The first part of the document also summarizes the main mediator components for the BIONETS service life-cycle and provides the description of required primitives and interfaces.

The second part of document concentrates to solutions from WP4 and SP2 which are applied to BIONETS service framework and mainly to the service composition and security.

For the service security we present the identity management with which it is possible to provide unique identifiers, names, for service individuals also globally. The service name is based on combination of service specific semantic description and secret/public key pair. Using the public key cryptographic system in disconnected BIONETS type of environment it is also needed to ensure that some compromised key pairs do not compromise the whole system. For this we also present the trust and reputation management and service configurations and execution management mechanisms for the service individuals and cells.

The trust and reputation management is used for monitoring the behaviour of the service in order to rate and evaluate its trustworthiness. The monitoring is carried out with frequent user feedbacks and/or specific parameters and identification of malicious services, which can be also done proactively. In practise the trust and reputation system provides the information of already running services but usually the system can be violated as soon as a services starts running (for example unauthorized processes can be executed etc.). Thus, processes which may conflict with installed security policies need appropriate assistance. For this we introduce also the service configuration and executions management, which main purpose is to prevent the executions of malicious, unauthorized services and to complement the trust and reputation mechanisms by ensuring that the configuration and execution of the service comply with the security policies of the system.

Respectively to WP4 we also introduce in this deliverable some fundamental studies from the SP2, which has an impact of the service life-cycle. The work applied from the SP2 concentrates on the development of a biologically-inspired interaction computing model and of its formalisation through a corresponding specification language. In the Section 3.2 we present some consideration for Lie group analysis of metabolic pathways as well as comparison and analysis of automata for service composition. The main outcome from the study is that automaton realisation of the service interaction machine is possible and can be attempted, with the objective of reproducing biological behaviour in computer science applications such as security and service composition.

Document History

Version History

Version	Status	Date	Author(s)
0.1	ToC	2009-09-04	Janne Lahti
0.2	Updated ToC	2009-10-15	Janne Lahti
0.3	First draft	2009-11-17	Janne Lahti, Paolo Dini, Silvia Elaluf-Calderwood,
0.4	First draft	2009-11-18	Janne Lahti, Heiko Pfeffer, Daniele Miorandi
0.5	Draft	2009-11-24	Janne Lahti, Silvia Elaluf-Calderwood, Daniel Schreckling
0.6	Draft	2009-11-30	Heiko Pfeffer
0.7	Draft	2009-11-30	Daniele Miorandi
0.8	Final editing	2009-12-1	Janne Lahti
1.0	Internal review	2009-12-3	Janne Lahti, Jyrki Huusko
1.01	Corrected version	2009-12-14	Janne Lahti
1.02	Corrected version	2009-12-18	Janne Lahti, Paolo Dini, Daniel Schreckling, Daniele Miorandi, Louay Bassbouss
1.1	Final version	2009-12-29	Janne Lahti, Jyrki Huusko

Summary of Changes

Version	Section(s)	Synopsis of Change
0.1	All	ToC proposal
0.2	All	Updated ToC proposal
0.3	All	Final ToC
0.4	2.1, 2.4, 2.5, 4	First drafts of sections
0.5	2.1, 2.2, 2.3, 3	First drafts of sections
0.6	2.4, 2.5	Update, restructuring Section 2
0.7	3	Final versions of sections
0.8	2.3	Final versions of sections
0.9	All	Final Editing
1.0	All	Internal review
1.01, 1.02	All	Editing
1.1	All	Final version

Contents

1. Motivation and Relevance	5
2. SP3 Achievements	6
2.1 BIONETS Service Environment	6
2.2 Biologically-Inspired Aspects in the BIONETS Service Framework	9
2.3 Overview of Service Framework Business Models	11
2.3.1 Socio-economic metaphors applied to the BIONETS service Framework	11
2.3.2 Evolving Business Models: Communication Exchanges the BIONETS case	12
2.3.3 A taxonomy of possible BIONETS services for the current service framework.....	14
2.3.4 Remarks	15
2.4 Service Framework Mediators	16
2.4.1 Mediators: BIONETS enabler	16
2.4.2 BIONETS Mediators at Work: An Example	23
2.4.3 Conclusion	25
3. Co-operation with SP2 and WP4.....	26
3.1 Overview of Service Security Framework.....	26
3.1.1 Challenges.....	26
3.1.2 Identity Management	26
3.1.3 Trust and Reputation Management.....	27
3.1.4 Secure Service Configuration and Execution Management	28
3.1.5 Conclusion	29
3.2 Algebraic analysis of service composition automata	29
3.2.1 Overview of algebraic and Lie group analysis of metabolic pathways	29
3.2.2 Timed automata vs. regular automata	31
3.2.3 Conceptual description of a model of Interaction Computing	32
3.2.4 Categories from Service Compositions	34
3.2.5 Conclusion	36
4. Assessments.....	37
5. References.....	39
Appendix A: Terminology	43
Appendix B: Service Migration Mediator	45
Appendix C: Summary of Some Elementary Finite Group Facts.....	47
Appendix D: Background for p53-mdm2 system	50
Appendix E: Lie Symmetry Analysis of p53-mdm2 System	52

1. Motivation and Relevance

The main goal for this document is to wrap up the work which has been carried out in the BIONETS Sub-project 3 and work packages WP3.2 and WP3.3 during the whole four project years. The document is not targeted to provide the service architecture definition such as the deliverables [BIONETS_D113/313] and [BIONETS_D114/314]. If anything, this deliverable extends the BIONETS service architecture definition by wrapping up the research results from BIONETS WP3.2 and WP3.3.

The main driver for us to provide the comprehensive documentation was that the work for service life-cycle, architectural components, bio-inspired techniques and socio-economic models had been distributed into several different deliverables and thus it is challenging to find the main solutions of our research and the BIONETS service ecosystem especially for the people who has not followed the project from the start.

The deliverable acts also as a “cookbook” for BIONETS service life-cycle and for all the required mediator components to facilitate the service autonomy. We have also defined the service management plane which enables the services to follow the BIONETS service life-cycle.

In addition we revise briefly the role played by biological inspiration in the design of novel techniques and mechanisms for the BIONETS service framework. The overall goal was to devise solutions able to enhance the autonomic features of BIONETS services. The role played by various techniques was mapped to the reference MAPE-K model from IBM. Three main lines of biologically-inspired research were pursued, based on the use of techniques mutated from Evolutionary Strategies, Chemical Computing and Artificial Embryogenesis fields.

We also present the advances in the analytical modelling to carry out the service composition. For this we have further developed the mapping of metabolic pathways by combining Lie group analysis of a cell regulatory pathway with the algebraic analysis of the semigroup associated with the automaton derived from the same pathway. This has given us interesting insights in how the cell may achieve self-organised behaviour and in the properties of automata that could achieve analogous behaviour. We further developed a category theory formulation that will enable us to understand the interdependence between automata structure and behaviour and that will help us develop a specification language for autonomic environments. One of the key elements of the service ecosystem and service life-cycle is the security framework which is also presented in the deliverable although the main development is done in BIONETS WP4. The work in the SP3 has also included the research of the components for service architecture that facilitate the sustainability of alternative socio-economic business models, which can in turn support new business models. The socio-economic metaphors applied to BIONETS service framework and the evolving business models are also presented in the deliverable.

The deliverable is constructed as follows. In Section 2 “SP3 Achievements” we go through and wrap-up the main features of the BIONETS service framework, SP3 and provide a summary of the main required components of the service framework – The Service Mediators. In Section 3 we provide the deeper insight to some specific approaches such as algebraic analysis of metabolic pathways and automata for service composition and security framework. The Section 4 provides the summary of the most essential research challenges for BIONETS evolving services and the solutions provided and finally Section 5 is for the conclusions.

2. SP3 Achievements

2.1 BIONETS Service Environment

In this section we will give an overview of BIONETS services and the environment where they exist. We will use top-down approach by first characterizing the envisaged BIONETS environment from where the services stem their unique requirements and functionalities. We give an overview of the developed SerWorks architecture and describe the Service Framework part more closely, as it is providing the execution environment for Services and Service Mediators. Finally we define the actual Service components, Service Cells and Service Individuals and discuss the life-cycle of services. A more detailed definition of these components can be found from WP3.1 deliverables [BIONETS_D311, BIONETS_D312, BIONETS_D113/313] presenting the BIONETS SerWorks architecture.

Services and Mediators

A pervasive computing and communication environments, such as BIONETS, can be characterized by an extremely large number of networked devices possessing rich computing and communication capabilities. In such environment we cannot rely on the availability of the backbone connection. The notion of traditional end-to-end connection will fade and the local connectivity will increase. Conventional approaches to communications and service provisioning are ineffective in this context. The devices do not form a connected topology but live in an ecosystem in which highly dynamic islands of connected nodes form and disappear continuously. Services operating in such environments are bound to be autonomic, and evolve to adapt to the surrounding environment. A Figure 1 below presents a BIONETS environment and its basic components. A more detailed description of BIONETS environment and its components can be found from [BIONETS_D311, BIONETS_D312].

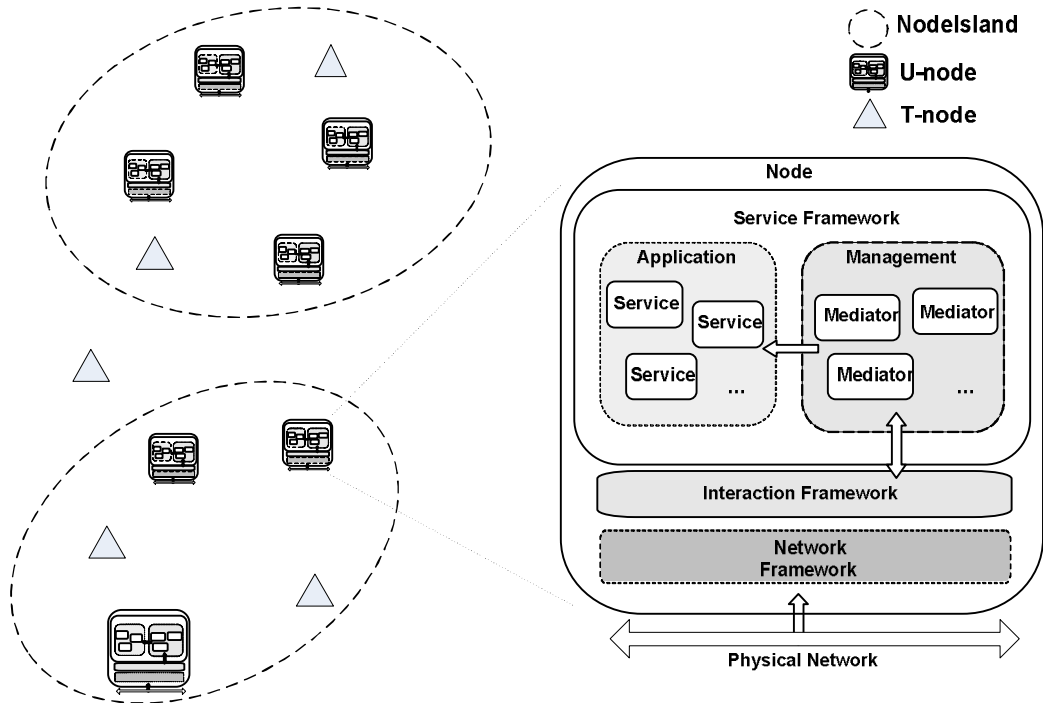


Figure 1: BIONETS Service Environment

SerWorks

SerWorks architecture, as presented in [BIONETS_D114/314] takes a unified view on networks and services. In order to tackle the problems with BIONETS environment the SerWorks architecture incorporates service and network architectures and benefits from the biologically inspired communication paradigms. The SerWorks architecture definition was first presented in “SerWorks Architecture v1.0” [BIONETS_D113/313] and later revised in [BIONETS_D114/314] by adding among other things a description of the impact to existing systems and application scenarios

Services and service management in BIONETS SerWorks concept are autonomic, and services evolve to adapt to the surrounding environment, like living organisms evolve by natural selection. Network operations will be driven by the services, providing an “ad hoc” support when and where needed to fulfil users’ requests.

The unified service-oriented approach of the SerWorks aims to enhance especially the flexibility of current network and protocol system by providing common methods for network and service domain interactions in a such a way that the service-tailored protocols can be built and adapted on-the-fly according to service and user requirements. In SerWorks, the network becomes a set of particular services, aimed at general connectivity purposes, which can be created and modified at runtime in order to accommodate for specific system conditions and higher-level constraints.

As illustrated in Figure 1, the core of the SerWorks architecture consists of three frameworks: the Service Framework in the upper layer of the architecture, the Interaction Framework in the middle layer, and the Networking Framework in the lowest layer. Networking Framework provides the basic communication capabilities in the disappearing network context. The Interaction Framework provides communication between Services, Mediators and Network Framework. The Service Framework, provides the execution environment for service components and mediators, and is discussed with more detail in the following section.

Service Framework

The main building blocks of the Service Framework are Service Cells (SC), Service Individuals (SI) and Service Mediators [BIONETS_D311, BIONETS_D312]. In order to implement node level autonomy in a flexible way, service mediators were introduced as architecture elements complementary to the services. Service mediators behave like agents for the services sharing the same node with them. Services and service mediators interact through the Interaction Framework, no matter where they are hosted.

The BIONETS service framework plays a two-fold function. On one hand, it provides a runtime environment for the execution of the service logic running on U-nodes and T-nodes. On the other hand, it includes the capabilities to react in an autonomic way to changes in the environment. These capabilities include different evolution and adaptation strategies, service life-cycle handling and mobility support [BIONETS_D321].

Service Mediator

The different Service Mediators provide the autonomic control functionalities, network connections and other platform resources for all Services located inside the same node. In this kind of node-level autonomicity, the logic providing all the autonomic functions needed in the service is located in the node hosting the services. This approach brings many benefits for the BIONETS Service architecture [BIONETS_D311]. For example, the Services can be designed to be more light-weight and the code duplication of cooperation-related processes on one node is minimized. Another benefit is that the services are kept simple; therefore, they may also be provided by T-Nodes and not just U-Nodes. The Service Mediators are autonomous applications mirrored on all nodes that need or wish to support a particular organization feature. For that purpose they may also be duplicated and distributed among nodes. Service Mediators are supposed to implement all aspects of autonomy that are not application-related and therewith form the dynamic glue logic holding the services together.

The different types of Service Mediators, their roles and functionalities are presented in more detail in the Section 2.4.

Service Cells and Service Individuals

In BIONETS the definition of service follows the OASIS service definition, which provides a reference model for service oriented architecture. [OASIS06, BIONETS_D114/314]

The service is a mechanism, which enables access to one or more system capabilities. The access is provided using a prescribed interface and is exercised consistently with constraints and policies as specified by the service description [OASIS06].

Unlike OASIS Service-oriented architecture (SOA), BIONETS does not distinguish between application- and network-level components from the management point of view, so in effect any network and application domain mechanism that fulfils the above mentioned requirements is considered a service within the SerWorks architecture i.e. the Service Cells, Service Individuals and Network Components which obey the service definition by default are treaded similarly in the system. In addition, in the SerWorks architecture, the services needs to follow strictly the various phases encompassed by the BIONETS service life cycle [BIONETS_D321] and provide an interface for the specified Service Mediator components.

BIONETS Services can be divided into Service Cells and Service Individuals. A Service Cell (SC) is an atomic service component running on top of the Service Framework and can be any kind of small application providing certain functionality to other services or users. As the Service Framework mediators provide all the autonomic management functionalities the service only needs to implement the basic life-cycle controlling interfaces for starting, stopping, migrating, replicating and deprecating the service. With these interfaces the mediators can control the life cycle of the services located in the node. [BIONETS_D312].

A Service Individual (SI) is a composition of Service Cells. Service Individuals may also be composed of other Service Individuals in a recursive way. Service Individuals are expressed with a graph representation [BIONETS_D323], wherein nodes represent types of Services (the elements of the composition); edges (links) represent the interactions (communication) between the Services. [BIONETS_D312].

Autonomic Service Life-cycle

According to the BIONETS concept services and service management are autonomic, and services evolve to adapt to the surrounding environment, just as living organisms evolve by natural selection. The purpose is to take the autonomic procedures in a level where the users will need only to make relatively less frequent predominantly higher-level decisions, which the system will carry out automatically via more numerous, lower level decisions and actions. Several similar autonomic service solutions have been introduced on other biologically inspired communication platforms. Nakano and Suda in [Nakano 2004] and [Nakano 2005], for example, have introduced a network framework based on software agents to model the services. In contrast in the BIONETS system the Service autonomicity is achieved at node-level by utilizing Service Mediator entities for management and decision-making. [BIONETS_D312].

With BIONETS the Service autonomicity is achieved mainly through different phases of Service life-cycle, controlled by Service Mediators. The Service life cycle is designed to allow services to emerge in an autonomic fashion in order to gain support for user tasks while reducing the user's efforts in service creation. This goal is regarded against the background of pervasive service environments, composed of services originating from the users' computing devices and those relevant in the users' context and their direct surrounding. [BIONETS_D114/314]

In BIONETS the main objectives of service life-cycle research are the *creation of service on demand of the user* and the *continuous adjustment of services* to cope with changing requirements in the environment. The *creation of service on demand of the user* addresses basics to achieve an applicable service environment at all, while *continuous adjustment of services* addresses the progressive improvement and adaptation of services in the presence of

highly dynamic computing infrastructures. According to the objectives, we defined three phases for the life cycle, *Preparation and Creation*, *Evolution* and *Integration*. The *Preparation and Creation* phase comprises activities around acquisition of user needs and environment context as well as the creation of functionally appropriate Services addressing. The *Evolution* phase describes all activities concerning improvement, adaptation and proactive creation of Services. Finally yet importantly, the activities of *Integration* phase address the actual handling of Services in BIONETS environment. [BIONETS_D114/314]

The BIONETS Service life-cycle is presented in detail in the [BIONETS_D321] and later refined in [BIONETS_D113/D314].

One factor of achieving autonomic service, not strictly part of but closely related to life-cycle, is Service mobility. The Service mobility offers solution for improving system efficiency, balancing load in the system and optimizing service penetration. The service mobility concept includes not only the migration procedure but also the selection of the service, possible replication or deprecation of the original service, migration of service implementation, its execution state and runtime data, adaptation to the target platform and handover of user associations. The proposed Service migration approach is presented in more detail in [BIONETS_D322, BIONETS_D324 and BIONETS_D325] and some initial results verifying the proposed approach are presented in [LRHK09].

2.2 Biologically-Inspired Aspects in the BIONETS Service Framework

In this section we review the role played by biologically-inspired aspects in the definition and refinements of the BIONETS service framework. In particular, we describe how appropriate bio-inspired techniques have been applied to reach autonomic properties and review the outcomes of our research in the field.

Three main bio-inspired approaches have found application in the design and management of BIONETS services: evolutionary computing [ES03], artificial embryogenies [SM03] and chemical computing [D04]. They have all been revised in [ADMS09, MYDP10]. While such approaches are probably not sufficient to realize the full vision of Autonomic Computing, [KC03] they can provide substantial contributions to the design of autonomic software, especially in terms of:

- Evolutionary computing approaches can constitute a basis for designing self-optimizing and self-configuring strategies, due to their inherent ability to search for optimal operating points over unknown fitness landscapes;
- Chemical computing can represent a suitable execution model for software undergoing run-time changes, due to its inherent robustness and native support of modifications during execution;
- Artificial embryogenies can be used to design self-healing mechanisms for distributed software as well as to enhance the scalability of evolutionary techniques.

In terms of the reference MAPE architecture [KC03], we can map the techniques we will overview as follows (see also Fig. 2):

- **Monitor:** it performs context sensing, logging events related to the functioning of the service under control and possibly gathering information from peer nodes using a similar service.
- **Analyze:** it evaluates the performance level of the current service according to the data gathered by the monitor unit; this can be seen - using an Evolutionary Computing lexicon - as estimating the fitness of the current solution (set of policies, rules, algorithms driving the behavior of the component and implementation thereof).
- **Plan:** given the fitness level estimated by the analysis unit and the information contained in the knowledge base, it operates changes on the service software structure, through, e.g., methods inherited from evolutionary computation (reproduction, variation, selection).

- **Execute:** starting from the new solution created by the planning unit it generates a new executable version of the software, replacing the current one. This process could encompass a developmental phase, during which the structure of the solution, as specified by the planning unit, is “grown” to build the executable service. During such a phase it is possible to make use of information available on the current context (gathered by the monitoring unit and stored in the knowledge base), in order to obtain solutions adapted to the current environment. The developmental phase can also be performed by tuning the “concentration” of various developmental rules, according to chemical computing paradigms.

The MAPE scheme decomposes the system into a small set of functional blocks: as a cautionary remark we notice that even though the IBM MAPE cascade is universally adopted as “the” reference scheme for autonomic systems, in general, the mapping of a practical system into the proposed architecture might be per se a non trivial task.

According to the SerWorks approach [BIONETS_D113/313] the methods devised can be applied to both network- and application-level services. In this deliverable, we focus on the latter case. More details on the application of bio-inspired techniques to the former case can be found in [MYDP10].

It is worth remarking that the introduction of biologically-inspired techniques in service frameworks required the adaptation of currently used models for service lifecycle [BIONETS_D321]. Novel bio-inspired models for service lifecycle were introduced in [LPRS07, PLRS07].

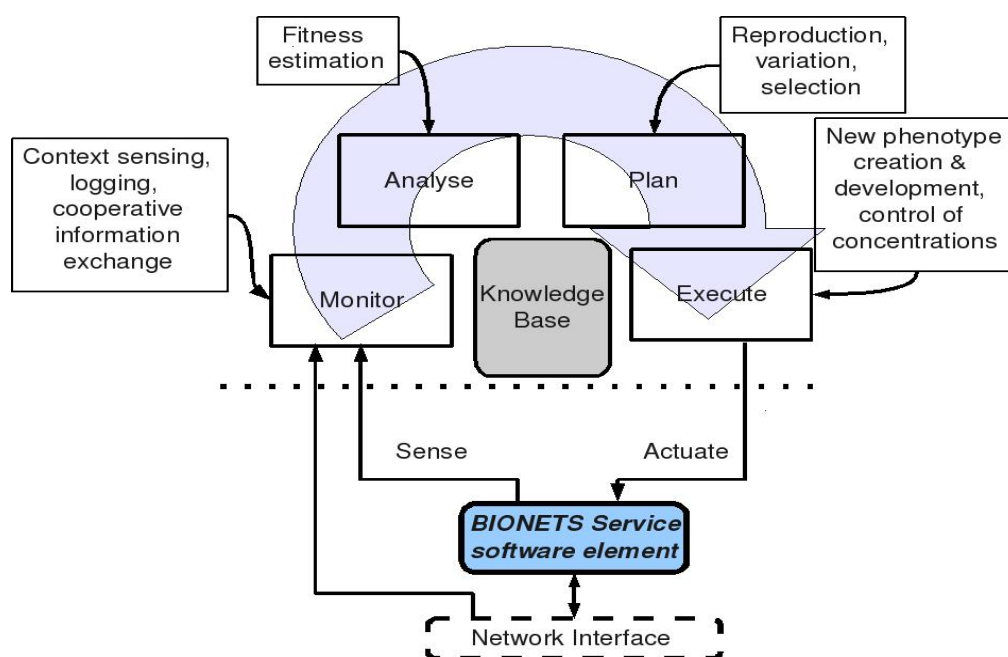


Figure 2: Application of Autonomic Computing MAPE control loop to BIONETS services and fitting of evolutionary, and embryogenic and artificial chemistries methods.

In terms of application of evolutionary computing techniques, the following results were achieved:

- For atomic services (Service Cells): dynamic configuration and optimisation of service parameters through application of cooperative mechanisms based on genetic algorithms coupled to gossip-based mechanisms for dissemination of information [CDPMW05, TMCDP09]. The methods showed to be able to converge to the optimal operating point in the presence of random meeting patterns among nodes and noise in the estimation process. Such results were reported in [BIONETS_D324].
- For composite services (Service Individuals): dynamic configuration and optimisation of service composition based on a mixed workflow-component approach by means of

evolutionary strategies [LPS07, PLS08, BHN09, BLH10]. The methods showed to be able to seamlessly enhance composite service by dynamically reconfiguring the composition graph as well as to re-configure after major changes (e.g., faults of a service cell) by means of genetic algorithms/programming. Such results were reported in [BIONETS_D324, BIONETS_D326].

In terms of application of chemical computing techniques, the following results were achieved:

- General templates for writing self-replicating and self-modifying programs in chemical computing languages (in particular: fraglets in particular) were developed [YSM07]. Ability to self-heal in face of deletion of arbitrary portion of code was demonstrated through the use of the so-called "autocatalytic quines", which are self-replicating programs able to perform some embedded computation [MSTY08]. A method for the optimization of individual service cells based on artificial chemistries was devised, based on our energy framework for resource control [MYBT09]; the result has been submitted for publication and is currently under review [Y10]. Such results were reported in [BIONETS_D228].

In terms of application of artificial embryogenies techniques, the following results were achieved:

- An architecture and a set of methods based on the concept of service stem cells for achieving self-healing behaviour was first presented in [MY08]. A refined version, including application to a distributed service (distributed network monitoring) and experimental results were presented in [MLY09]. Application to BIONETS was reported in [BIONETS_D326]. The proposed system showed to be able to handle faults (at both the level of nodes and software) in an effective and scalable way, being able to re-establish a valid configuration in short times and in a totally distributed manner.

2.3 Overview of Service Framework Business Models

2.3.1 Socio-economic metaphors applied to the BIONETS service Framework

The use of mobile devices and the 'Web 2.0' phenomenon have opened virtual and real windows for ubiquitous and seamless social communications [AGA04], which are paradoxically both continuous and fragmented [WEI07]. Although many research efforts have focused on understanding the social aspects of networking using mobile devices [BS08], the understanding of how these networks can be valued in terms of money is an unfinished research topic. Social networking draws its foundation from Granovetter's theory built around the concept of "the strength of weak ties" [GRA82]. The brilliance of this theory helps understand how some individuals can be nodes of convergence for many other individuals that otherwise would have nothing in common [BAR02].

In many ways the use of mobile devices has taken the lead role in introducing new social organizational forms [RHE02]. The combination of mobile devices with networking tools such as Facebook, Twitter, Hi5, and other similar web-based applications – collectively referred to as Web 2.0 – has created a chorus of approval from social experts [QUI06, VAN09] for the apparent success of social networking as a means to achieve a ubiquitous connected society [GB0901]. Social networking in fact shifts the use of such media from the technology to the user and the content users generate [BAR02, HAR08].

This shift has been accelerated by the rapid evolution of mobile devices that are adopted by users, because these devices have features that can perform and enhance seamless communications. Users are eager to transfer and exchange many forms of data on the virtual networks accessed through mobile devices [SCH09]. Some of these communication exchanges open up new social networks; others reinforce already existing ones [WIR06].

Although this exchange of data (e.g. user profiles, video files, music files, documents, games, etc) benefits social networking, in economic terms the increase in volume of data transferred

has a cost that operators quantify. The emergence of pervasive and interconnected computing devices that give rise to a dynamic network topology in which distributed content can become focalized and spontaneously shared brings new opportunities for economic exchange. The patterns of communication are based on social networking and bio-inspired models such as epidemic spreading and gossiping metaphors [GH06, JEL05], where trust is all important and reputation keeps local trust values above other values in the network [BG98]. Thus, social networking and mobile applications have created a new real and virtual space in which the traditional models of revenue might need to be re-thought [MAK09, PRI07].

Telecom operators are the main referees when applying a distribution model to determine these costs. However in pervasive and distributed agent networks a significant volume of data transactions is distributed by and between nodes and not necessarily linked primarily to the telecom operator's backbone.

As users turn to providing more and more content, agent networks become an interesting case of study from an economic point of view. The value or cost for these distributed communication exchanges are not well quantified under current business models.

Current business models base their revenue calculations on conventional advertisement. The fact that this is an emergent technology is responsible for the dearth of economic studies looking into this subject. There is a wide berth between theory and applications. Current attempts to estimate these values have been undertaken by e.g. charities trying to estimate the value of using social networking for their campaigns [FRO07]; others rely on models in which each user assigns the relative value to the communications within the network, as Twitter users discuss [PRI07, SCH09, TWI09].

In WP3, the business modeling has been focused in the case of one of these networks and the characterization of its potential value of such networks. For this purpose Benkler's [BEN05, BEN06] proposal about value-added distribution and the emergence of sharing as a mode of production is used as the basis for the simulation which was developed within the scope of this project, while focusing on the economics of sharing for the distribution of content. The idea behind this simulation is to provide a benchmarking tool that can support the development of distributed applications over a disconnected network of mobile devices and sensors within the context of the BIONETS EU project.

2.3.2 Evolving Business Models: Communication Exchanges the BIONETS case

A network such as BIONETS, where the emphasis on communication exchanges and engagements is influenced by an evolutionary bio-inspired framework, that is node-based and distributed, is an ideal environment where to try to determine the values of communication exchanges and how to quantify them. In many ways the BIONETS case fits the requirements as many of its processes are seen as an industrial mutation that incessantly revolutionizes the economic structure from within, incessantly destroying the old one, incessantly creating a new one [BIONETS_D332, SCH09].

Figure 3 shows the primary and secondary actors that can be found at all layers of the BIONETS architecture. The actors aim to facilitate communication exchanges of diverse types, and consider for those five main components:

- a) Technology expressed by the boundaries determined by device manufacturers and network equipment vendors.
- b) Services: the discussion in this document focuses on value added services, content and applications that users or other network devices can access through the BIONETS mobile network: the symbiotic relationship between content providers (individual or networked), application providers, and payment agents/or exchange agents.
- c) Network, U-nodes, T-nodes that effectively work at the nuclear level as mobile network operator or ISP

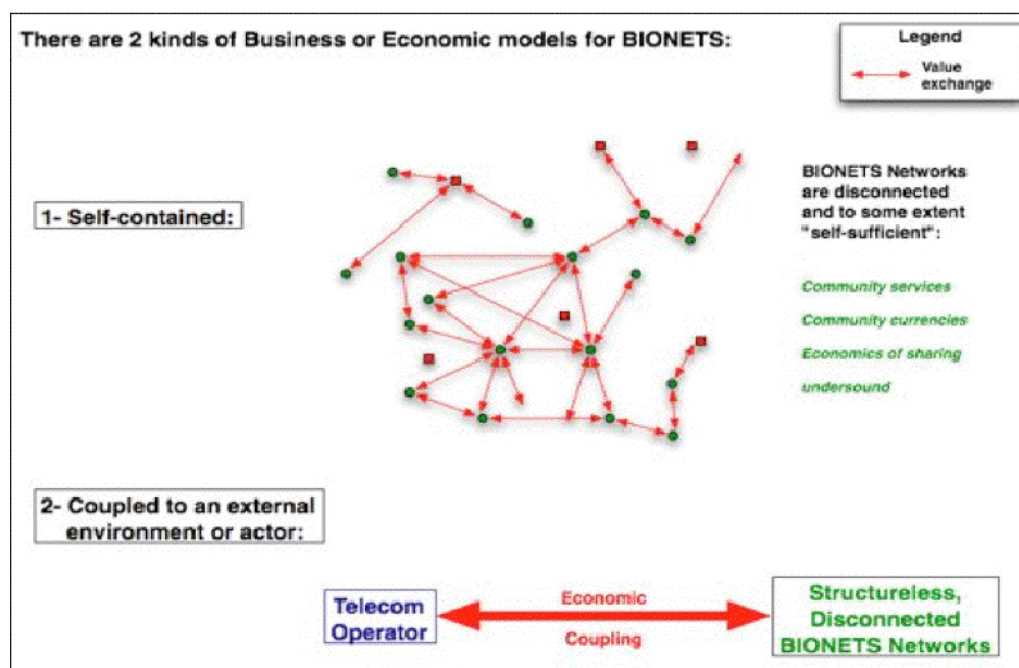


Figure 3 – Actors on BIONETS Networks

In the context of BIONETS there are a number of economic paradigms that need further explanation in order to understand the working model for the Service Framework Business models. At the core of the work on usage scenarios presented by several BIONETS partners there are two main networks of interaction in which business models might develop. The first network is a self-contained, 'disconnected' and distributed BIONETS network of nodes (actors) able to exchange data with each other using the token values of alternative trading systems – where no actual money changes hands – whilst the other network is that used by the telecom operators, in which economic revenues are based on more conventional business models. Both models are depicted in Figure 3.

Neither of these networks has an exchange rate to convert the services or transmissions within the distributed BIONETS network into revenue, nor within the conventional telecom network. In WP3, the decision to build a computing model using a modelling tool to illustrate alternative business models is a way of evaluating how these exchange rates could be calculated.

The economic principles of the model are based on the use cases developed by the BIONETS consortium, which can be grouped at an abstract modelling level as the ability of each node or actor to transmit (send or receive) up to five types of data (this can be changed in the model), assigning to each of these transmissions a token value for the desirability of this transaction. The data transmitted can be for example either environmental variables such as the ones presented in the Digital City scenario [BIONETS_AS07], with services that for example allow real-time access to Digital Maps or music exchanges, or a combination of all of these.

A generic model in this case tries to express the added value of the exchanges that occur in the node-based networks that are traded, for either other tokens or real monetary value, by the actors or users of the network. To illustrate this idea, consider the situation in a metro station in which any agent in a certain area within the station can run BIONETS applications. Each agent can receive or send data based on their needs. Assume that at some point in time there is a number of people waiting on the platforms for trains to arrive; whilst waiting, some if not all of those people might use their mobile devices locally. Each of those devices is an agent in the business model being simulated. As agents discover other agents they might start exchanges of data that result in economic transactions based on agreed token values or evolving token values. A historical record of such transactions could be stored in an

individual virtual account for each agent, thereby making possible a valuation of the desire or ability to exchange data successfully.

One type of exchange could be the case of an agent broadcasting to other agents within range a message or data file containing some kind of advertisement; all the other users within range might choose to accept or reject the sender's file; if, however, an agent decides to accept the sender's file, the sender is credited with a token value paid into their virtual account. This could be the case for a localized advertisement used to reach a small network of users: the eagerness of users or agents to accept the advert could ultimately be converted into real monetary units for the sender by, for example, the telecom operator providing the permanent network.

Over time the number of agents in the metro station changes; at some times there will be peaks of data exchange and/or number of agents, and at other times minimum or no exchange of data or no agents at all. There is a dynamic cycle, based on agents entering and leaving the metro station, wishing to exchange or trade information. Some of the evolving, bio-inspired applications from BIONETS will merge then with social networking behaviour to express over time the changing nature of these transactions.

Conventional business models cannot effectively allocate value to this type of agent network in which the exchange of data is or can be considered separate from the backbone telecom network, as is the case for example with Bluetooth. One of the aims of the research completed is to illustrate how alternative economic models can actually build up enough subjective or token-based value to make it worthwhile to develop an exchange rate for its conversion into real money.

This can be done by evaluating the total of the exchanges in the metro station, in terms of both the number of tokens and the volume of data transferred, against the telecom's valuation of volume data transfer per minute. This will allow for example potential marketing companies and telecom providers to estimate, based on the potential number of users or actors accepting a broadcast message in the metro station, the cost of advertising localised and perhaps focus-orientated advertisements.

2.3.3 A taxonomy of possible BIONETS services for the current service framework

The purpose of this proposed taxonomy is to classify the economic areas in which BIONETS services and applications – based on the current service framework -are strong enough to provide sustainable business models by using an alternative business approach to the generation of revenue. It is envisaged that there are six major areas for expansion and development. These can be contextualised within the many applications or services that could exist within the Digital City use case umbrella.

A first category is the standard free access. BIONETS applications (software and services) can be used, downloaded or distributed for free. It could be the case that some content in a basic form will be provided for free too. This “free” service is provided or accessible to all users of the basic or shell versions.

A second category is advertising. Users can access content, services, software and more in exchange for visualising or interacting with some form of advertisement. The advertising sponsors then cover operational costs of the network. Ideally this category will be “free” to everyone.

A third category is cross-subsidies. This category has been included in BEBS software and includes any product (software, service and content) that entices the user to pay for something else. An exchange rate can be external (e.g. by the telecom operator) or internal (e.g. through the collection of eBay points) quantified. It is only perceived as “free”, however, as it depends upon success of products or services that most users will eventually be willing to pay for, one way or another.

A fourth category is zero marginal cost. This category has been included in the software services and includes products (software, service, and content) that can be distributed without

an appreciable cost to anyone. This is possible due to the intense level of access to such products. This service ought to be free to all users.

A fifth category is labour exchange. This category is drawn from Internet history. The transition to mobile is well in execution and it is sensible to forecast a transition of web sites and their services to the mobile arena. The “free” service is accessible to all users, since the act of using these sites and services actually creates something of value.

A sixth category is the gift economy. This category is inspired by the free-for-all models of the Open Source movement. Envisaged as an open source software or user-generated content, the “free” service is accessible to all users.

2.3.4 Remarks

The work completed in WP3.3 by the business units had in principle addressed a first attempt to develop a quantification of business models for the BIONETS infrastructure. The aim is to be able to provide, in the longer term, a model that can be interfaced with the computational output from the infrastructure development, feeding in this way real data collected in transactions that will occur when there are applications available to be deployed using the outputs from the BIONETS computational and scientific research. This is a novel approach that could be developed further and merged or integrated with the work completed by other partners in the BIONETS project. In doing so, WP3.3 can be a powerful tool to illustrate the economic benefits for actors derived from the exchange type of business model in any future BIONETS-enabled environment.

Until now the value of these trading networks has not been tested in mobile environments, and depending on their growth and sustainability some of these networks may in future evolve to have significant value, which will make them attractive to conventional sources of funding. E.g.: record companies releasing songs over metro stations, broadcasting companies (TV, film) distributing total or partial media files as teasers for users to develop interest in the content, users who can dedicate their resources to collect environmental variables that can be used later to tailor services or activities in a location according to user demand.

There are not many research computer models especially created to recreate social networks' economic behaviour, such as the ability to exchange information over distributed networks and allocate token values to the exchange transactions. The BIONETS network is perfect incubation for a business model that is not centralized; hence the research completed within 3.2 contributes to enhance the understanding of these types of networks, examples of this type of work can be found in the work of Harwood [HAR08].

The added value of this type of networks seems to be subtle [VAN09]. The added value is what makes these networks an interesting opportunity for alternative business models based on individual allocation of access to more data. It is able to view more data on the user, some of which may be intentionally obscured from the public or strangers. This allows users to network with a specific user in a more intimate and personal setting. Furthermore, it facilitates the creation of greater communication options, which, depending on the social site, opens up new avenues of communication. This adds a greater level of interactivity: you can connect with the person who added you through private/direct messages, instead of the highly visible public channel.

Since users are able to recommend content, when someone adds someone else as a friend (and vice versa), activity or actions on the site may be recommended or ‘pushed’ towards the other user in some part of their administrative panel or profile. This means that users achieve greater automatic visibility whenever they use the social networking tool. And finally there is a great social verification, emerging from the auxiliary advantage of having many fans on social media networking tools. This social verification arises especially when there is some kind of self ranking from the users according to the number of followers/subscribers using the tool. Popular and visible users tend to accumulate friends more easily than unknown users, and can potentially be hubs for data distribution.

2.4 Service Framework Mediators

The mediators of the service framework encapsulate management and control functionality to support the BIONETS service life-cycle introduced in deliverable D3.2.1 and refined in D3.2.4 [BIONETS_D321][BIONETS_D324]. Thus, mediators provide functionality that supports a certain degree of autonomy, self-awareness, and self-control of services. By integrating the functionality provided by the single mediators into the single services, even services itself can follow the bio-inspired service life-cycle without the involvement of managing entities such as mediators.

As the deliverables in the scope of SP3 of the BIONETS project have been subjected to specific aspects of the service life-cycle introduced in the first deliverable of this sub-project, the mediators can be considered as the final outcome of SP3 with regard to a novel life-cycle support that enables services to evolve over time to remain responsive to environmental changes.

Section 2.4.1 gives an overview of the mediators that have been created in the scope of the BIONETS project and provides –beside a short description– an interface specification make use of the mediator’s functionalities. The work of WP5 builds upon these definitions and is considered as proof of concept implementation of the mediators’ specifications. Section 2.4.2 outlines a small use case that involves several mediators and demonstrates their role within the BIONETS service life-cycle. The use case itself is also implemented within the scope of WP5 and will be part of the final BIONETS demo.

2.4.1 Mediators: BIONETS enabler

This section discusses the mediators of the service framework including their internal logic, interfaces and specification of possible messages or data exchanged among multiple mediators. Each U-Node includes a custom set of mediators which can interact with each other by calling the mediators’ public interfaces. Some mediators can be triggered directly from the end user, or more precisely from the user interface. Other mediators can also notify the user interface and update the user frontend by sending notification messages or by triggering listeners registered by the associated mediator. Mediators on different nodes cannot interact directly with each other. The only possibility to trigger mediators on remote nodes is to use the publish/subscribe mechanism over the opportunistic network. In this case, one node (publisher) can publish data or services in the network and another node (subscriber) can subscribe for special kind of data or services by the network. The network then triggers the associated mediator on the subscriber node side.

Figure 4 shows a simplified architecture of a U-Node including the *user interface*, *service framework* and *network interface*. The user interface allows the bidirectional interaction between the user of the associated U-Node and the service framework. The user can interact with the service framework by triggering the request mediator over the user interface. Other mediators of the service framework can also trigger the user interface; for instance, the service cell execution mediator can update the user frontend during the execution of a service cell if required. Furthermore, the network interface acts as a bridge between the service framework and the opportunistic network. Some mediators such as the discovery mediator or the service migration mediator can use this interface to subscribe for a special kind of services or for publishing available local services in the opportunistic network. The service framework also includes two repositories for storing services and data available on the current U-Node as shown in Figure 4. All public services are published in the opportunistic network and are visible for other U-Nodes or T-Nodes. *Private services* are only visible on the local node. The same applies to the data repository: public data are automatically published in the opportunistic network and are a part of the shared data space (SDS). *Private data* are visible only for the user of the associated node.

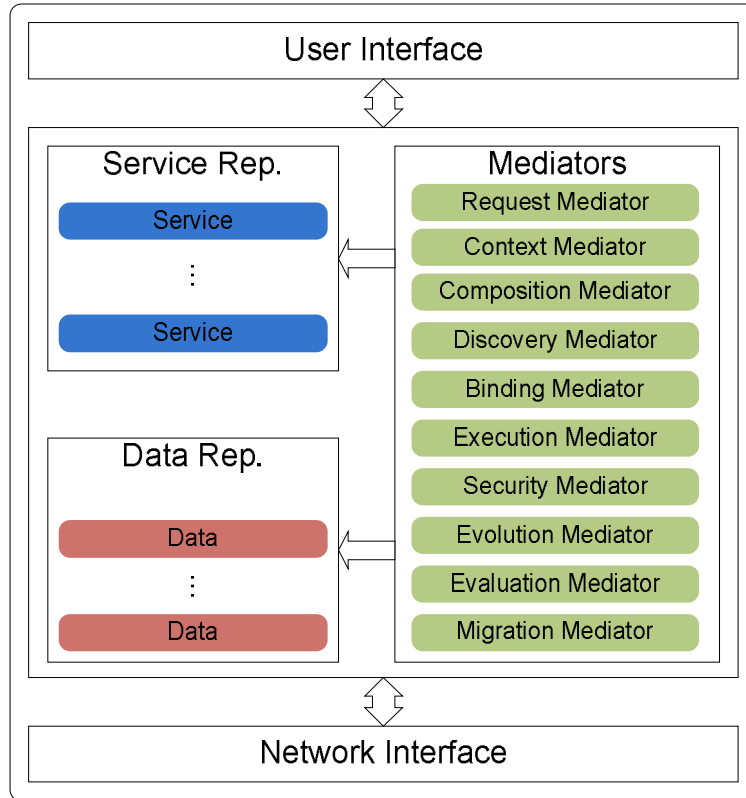


Figure 4: U-Node architecture

The mediators presented in the remainder of this section are considered as the mandatory to support services in following the bio-inspired service life-cycle as defined in deliverable D3.2.1 [BIONETS_D321]. Each mediator is shortly described; if possible, a reference to a BIONETS deliverable is given where research results related to the mediator's functionality are given. In addition, primitives for the access of mediators are specified, where the mediators or software components that commonly invoke these primitives supports services following the BIONETS service life-cycle are listed as *triggers*.

Table 1 shows the data types that are passed between the mediators; the specifications of the single mediators are outlined in the following sections.

Data Type	Description
<i>ServiceReq</i>	Represents the user service request in human readable form. Example: "Take a picture and publish it on a map in my current Location"
<i>ServiceRsp</i>	Represents the service response, which contains a suitable atomic or composite service to a given "service request"
<i>DataReq</i>	Encapsulates information about the data to search. Example: [content: 'video', tags: 'Berlin']
<i>DataRsp</i>	Contains data items suitable to a "data request"
<i>DiscoveryReq</i>	Contains semantic information about desired services. Example: "Services with outputs [Latitude, Longitude] and input [address]"
<i>DiscoveryRsp</i>	Contains discovered services suitable to a given "discovery request".
<i>CompReq</i>	Contains the same information as in <i>ServiceReq</i> but in computer readable form.
<i>AbstractPlan</i>	Represents composite non executable service.
<i>UserProfile</i>	Provides information about user preferences.
<i>DeviceCapability</i>	Describes features provided by the user device.

<i>ExecutablePlan</i>	Represents a composite executable service after late binding process.
<i>ServiceCell</i>	Represents an atomic service (Abstract + Executable Service)
<i>InputParam</i>	Represents an input parameter.
<i>OutputParam</i>	Represents an output parameter.
<i>ServiceIndividual</i>	Represents a composite service (Abstract + Executable Plan)
<i>Service</i>	Generalization for <i>ServiceCell</i> or <i>ServiceIndividual</i> .
<i>SimLevel</i>	Similarity level used to compare services during the evolution process.
<i>Activity</i>	Stands for a certain phase of the service life-cycle, such as “SC Execution”.
<i>Property</i>	Denotes a service’s non-functional property such as “response time”.
<i>ActivityGraph</i>	Encapsulates all collected measurements for a given <i>Activity</i> and <i>Property</i> as a graph.

Table 1 Data Types**2.4.1.1 Request Handling Mediator**

The *request mediator* represents the bridge between the user and the service framework of the U-Node associated to this user. The user request contains a formal description of the goal the user wants to achieve using human readable and understandable vocabulary. The job of the request handling mediator is to convert the user request from human- readable form to computer readable form, which the service framework can understand. The interface of the request mediator provides two methods for handling service and data request as listed in Table 2.

Primitive	Description	Trigger
<i>handleServiceReq(ServiceReq): ServiceRsp</i>	Receives and handles user service request	User Interface
<i>handleDataReq(DataReq): DataRsp</i>	Receives and handles user data request	User Interface

Table 2 Request Mediator Interface

The method *handleServiceReq* accepts a service request *ServiceReq* as input representing the user’s goal in human readable form. After converting the request in computer readable one, the request mediator delegates the new request to the composition mediator. The composition mediator tries to create a composition as an abstract non executable workflow, which will be forwarded in the next step to the binding mediator in order to create an executable workflow, i.e. service individual. Finally, the request mediator triggers the callback function with the service response *ServiceRsp* as input, which encapsulates the created service individual. Note that all methods of the request mediator interface are asynchronous, which means that the output of each method of the interface is the same as the input of the callback function.

The second method *handleDataReq* is used to find data items specified through the input parameter *DataReq*, which contains a detailed description of the desired data items. At first, the mediator search for corresponding item in the data repository on the local U-Node. If the mediator doesn’t find any item in the local repository, it subscribes for data with a search term generated from the input *DataReq*. Finally, the mediator encapsulates the found data items in a *DataRsp* and triggers the callback function with *DataRsp* as input.

2.4.1.2 Discovery Mediator

The discovery mediator operates on the service repository and searches for services which fulfill either the complete or a part of the given request. Table 3 shows the interface of the discovery mediator, which exposes the single method *handleRequest*.

Primitive	Description	Trigger
<i>handleRequest(DiscoveryReq): DiscoveryRsp</i>	Handles a discovery request to find new services	Composition Mediator

Table 3 Discovery Mediator Interface

handleRequest receives a discovery request *DiscoveryReq* as input, which contains semantic information about the services to search. Note that the service repository contains two kinds of services, namely abstract and concrete ones. Abstract services provide semantic information about the associated concrete services as IOPE (Inputs/Outputs/Preconditions/Effects) descriptions [BIONETS_D326]. Concrete services in contrast are executable services that are associated to an abstract service. The discovery mediator operates only on abstract services and compares the *DiscoveryReq* with each abstract service. The *DiscoveryReq* contains a snapshot of all open inputs and preconditions and available output and effects for a partial plan during a composition step. *DiscoveryRsp* encapsulates all found abstract services in a single object.

2.4.1.3 Composition Mediator

The composition mediator plays a central role in the service framework architecture. In case there is no single service that meets a given user request, the composition mediator tries to create a composition of existing simple services, i.e. service cells, to provide the required functionality by means of a set of services. The composition mediator operates only on abstract services and creates an abstract non executable plan. Table 4 shows the interface of the mediator, which provides the method *handleCompRequest* for handling composition requests.

Primitive	Description	Trigger
<i>handleCompRequest(CompReq): AbstractPlan</i>	Creates an abstract composition for a given request.	Request Mediator

Table 4 Composition Mediator Interface

handleCompRequest accepts parameter composition request *CompReq* as input, which denotes the internal representation of the user's service request (input of *handleServiceReq* in the request mediator). The composition mediator implements the composition algorithm based on heuristics and IOPEs [BIONETS_D326]. The mediator creates its first composition context from the given *CompReq*. The composition context is represented as a partial plan, which contains some open inputs and preconditions. During each composition step, the algorithm operates on a new partial plan retrieved from an old one by adding a single service returned from the discovery mediator. If there are no services on the local node that can be added to the current partial plan and the composition pool is empty, i.e. there are no more partial plans that can be extended during the next composition step, the composition mediator suspends the current composition and subscribes to the network interface for services using a search term generated from the current partial plan. In case new services are available in the environment, the network interface sends a notification to the migration mediator, which migrates the new services on the local device and sends a signal to the composition mediator to resume the suspended compositions. The composition mediator returns an abstract plan *AbstractPlan* representing the last partial plan with the last added abstract service as output.

2.4.1.4 Context Mediator

The context mediator provides an interface to get user profile information and device capabilities, which can be used from other mediators like the composition or binding mediator to control the composition or binding process, respectively. For instance, if the user profile indicates that the user prefers to use Google as map provider and the service repository provides two map services from both Google and Yahoo, the binding mediator selects the

Google Maps service. The same applies to the device capabilities; the composition mediator can only use services which require a GPS module (modelled as precondition) if a GPS receiver is integrated on the device of the current U-Node or on other node in the environment, which provides the same service. In this case the service can be executed remotely on the remote device. Table 5 shows the interface of the context mediator. This interface provides two methods *getUserProfile* and *getDeviceCapabilities*, which allow other mediators to access profile information of the current user or to retrieve features provided by the current device.

Primitive	Description	Trigger
<i>getUserProfile():UserProfile</i>	Returns the profile of the associated user	- Composition Mediator - Binding Mediator
<i>getDeviceCapabilities():List<DeviceCapability></i>	Gets the capabilities of the associated device	- Composition Mediator - Binding Mediator

Table 5 Context Mediator Interface

2.4.1.5 Binding Mediator

The key task of the binding mediator is to generate an executable plan from the abstract plan created by the composition mediator. During the binding process the mediator tries to find a concrete executable service for each abstract service of the composition. In case there are multiple executable services associated with one abstract service the binding mediator can make use of user profiles and device capability information provided by the context mediator to select the most suitable one. The result of the binding mediator's execution is an executable plan *ExecutablePlan*, which can be directly run by the execution mediator. Table 6 shows the interface of the binding mediator, which provides the method *handleBindingRequest* for handling binding requests.

Primitive	Description	Trigger
<i>handleBindingRequest(AbstractPlan):ExecutablePlan</i>	Creates an executable composition by replacing each abstract service with an executable one	Request Mediator

Table 6 Binding Mediator Interface

2.4.1.6 Security Mediator

The security mediator operates on the executable plan generated by the binding mediator. It returns a modified version of the given executable plan as output by adding new security constraints. Therefore, the security mediator accesses the security service repository during the modification process of the original executable plan, which holds references to all locally available security services, e.g., authentication and key exchange protocols etc., as well as security service cells, e.g., encryption function, random number generators, hash functions etc.

Primitive	Description	Trigger
<i>addSecurityModifications(ExecutablePlan):ExecutablePlan (modified)</i>	Adds security restrictions to a given executable plan	Request Mediator

Table 7 Security Mediator Interface

Table 6 shows the interface of the security mediator, which provides the method *addSecurityModifications*. The output parameter is the modified version of the given executable plan after security constraints have been added.

2.4.1.7 Execution Mediator

Depending of the kind of services to execute, the execution mediator provides two interfaces for each service cell (SC) execution and service individual (SI) execution.

SC Execution Mediator - The service cell execution mediator is responsible to execute single services, i.e. service cells. The realisation of this mediator depends on the kind of the service cells to execute. For example, if the browser is used as runtime environment, the SC execution mediator can delegate SC execution request to the JavaScript engine of the browser. In case of remote or third party calls, the SC execution mediators delegates the service call to the remote node or third party provider using a description file of the remote service interface. In case of REST services [FIELDING00], the Web Application Description Language (WADL) can be used to describe the service interface; the SC execution mediator then creates a REST call directly from the WADL file.

Primitive	Description	Trigger
<i>executeSC(ServiceCell, List<InputParam>): <OutputParam></i>	Executes a primitive service	- SI Execution Mediator - User Interface

Table 8 SC Execution Mediator Interface

Table 8 shows the interface of the SC execution mediator. The method *executeSC* accepts a service *ServiceCell* and a set *InputParams* containing the values of the input parameters as input and returns the set *OutputParams* containing the values of the output parameters, i.e. the results of the service's execution.

SI Execution Mediator - The service individual execution mediator is responsible to execute composite services or service individuals built from multiple service cells by means of additional control flow elements. The realisation of the SI execution mediator depends on the control flow language used to define service individuals. In the scope of BIONETS, timed automata based workflows are used to define service individuals [BIONETS_D326]. However, other realizations for composite services are possible. To execute the service cells the SI execution mediator delegates the SC execution requests to the SC execution mediator by calling the method *executeSC*.

Primitive	Description	Trigger
<i>executeSI(ExecutablePlan, List<InputParam>): List<OutputParam></i>	Executes a composite service	User Interface

Table 9 SI Execution Mediator Interface

Table 9 shows the interface of the SI execution mediator. The method *executeSI* accepts a bounded service individual *ExecutablePlan* and the set of parameters *InputsParams* containing the value of the initial input parameters as input and returns the outputs of the SI's execution.

2.4.1.8 Service Migration Mediator

The service migration mediator (SMM) allows the migration of services between multiple devices. The SMM is responsible for monitoring the service and node environment and triggering migration when needed. The SMM observes the overall state of the whole Node Island. It collects data from other mediators in the same node and also communicates with other service migration mediators in the same Node Island for getting all the relevant information to make rational decisions for migration purposes. The SMM performs the fitness evaluation based on the network information, node statistics (context and routing/forwarding information) and controls and initiates the migration process.

The actual decision and control logic are implemented using utility function based on the measurements monitored from CPU load, battery consumption, network connectivity and

service fitness (user evaluation, number of requests, etc.). The decision process is presented in more detail in Appendix B.

Primitive	Description	Trigger
<i>migrate(Service)</i>	Migrate new service on the local device	Network Interface

Table 10 Service Migration Mediator Interface

Table 10 shows the interface of the service migration mediator, which exposes the single method *migrate* to migrate the given service *Service* in the service framework.

The interface of this mediator is triggered directly from the network interface when a new required service is available in the network, i.e. in case the service framework has already been subscribed for services from the same category as the received service. After migrating the new service in the service repository, the migration mediator triggers the composition mediator to signal that new services are now available in the service repository. In this case the composition mediator resumes all suspended compositions that exist.

2.4.1.9 Observation Mediator

The observation mediator collects measurement data for each service in the service framework. The measurement data depends on the observed activity and property. For instance, by observing the SC execution the observation mediator can measure the response time or the failure rate each time a service is called. The same applies to the SI execution.

Primitive	Description	Trigger
<i>startObservation(Activity, Property)</i>	Starts the observation for a given activity and property	- all
<i>getActivityGraph(Activity, Property, TimeInterval): ActivityGraph</i>	Gets the graph for a given activity and property	- EvaluationMediator

Table 11 Observation Mediator Interface

Table 11 shows the interface of the observation mediator. The method *startObservation* starts the observation for a given *Activity* and *Property*. Here, *Activity* can stand for a certain phase of the service life-cycle, such as “SC Execution”, while *Property* denotes a service’s non-functional property such as “response time”. The method *getActivityGraph* returns all collected measurements for a given *Activity* and *Property* as a graph.

Note that multiple instances of the observation mediator are required in case multiple non-functional properties of a service should be considered.

2.4.1.10 Evaluation Mediator

The evaluation mediator uses the activity graphs generated by the observation mediator and the user profile from the context mediator to evaluate a set of services; this procedure is referred to as service evaluation and has been introduced in deliverable D3.2.5 [BIONETS_D325]. Table 12 shows the interface of the evaluation mediator, which exposing the method *evaluateServices*.

Primitive	Description	Trigger
<i>evaluateServices(List<Service>): OrderedList<Service></i>	Evaluates a list of services and returns an ordered list of services	- Binding Mediator - Composition Mediator

Table 12 Evaluation Mediator Interface

2.4.1.11 Evolution Mediator

The evolution mediator provides two methods *mutate* and *crossover* as shown in Table 13. The mutate operator produces a new *ServiceIndividual* from the original *ServiceIndividual* with a similarity level *SimLevel* (in [0..1]). The new SI has the same or a similar functionality as the origin SI, but, but may consists of a different set of service cells and thus might provide a different behaviour of the SI with regard to non-functional properties.. The same applies to the *crossover* operator, where two SIs are combined to a new SI with the same functionality.

Primitive	Description	Trigger
<i>mutate(ServiceIndividual, SimLevel): ServiceIndividual (new)</i>	Mutates a service individual and generates a new one with the same or similar functionality	- all
<i>crossover(ServiceIndividual, ServiceIndividual, SimLevel): ServiceIndividual (new)</i>	Crosses two service individuals and generates a new one with the same or similar functionality	- all

Table 13 Evolution Mediator Interface

2.4.2 BIONETS Mediators at Work: An Example

This section exemplary demonstrates the interaction between mediators running on multiple devices representing the U-nodes. As depicted in Figure 5, three U-Nodes are involved in this use case example. Each U-node provides a subset of the mediators introduced in the previous section as well as a custom set of service cells. Table 14 shows the available mediators, initial service cell available on each U-node, as well as the capabilities of the associated devices like “Camera” or “GPS”.

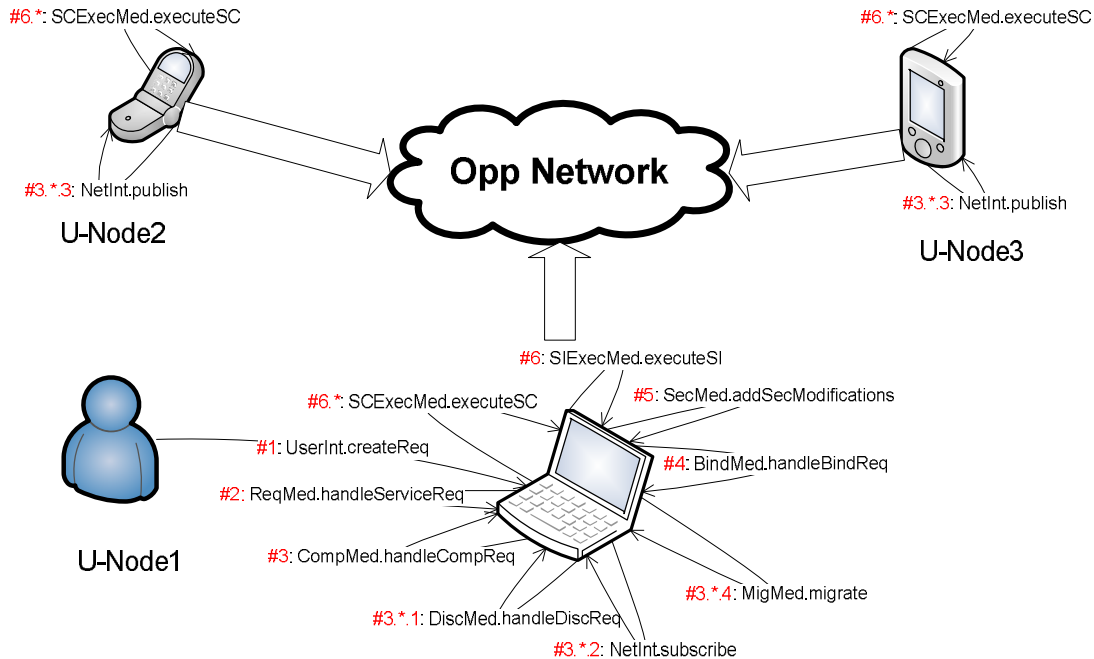


Figure 5: Use case example

The user of U-node1 creates the following human readable request (*ServiceReq*) using the user interface [#1]: “take a picture and show it on a map based on my current location”. Afterwards, the user interface calls *handleServiceReq(ServiceReq)* on the request mediator [#2]. The request mediator first checks whether there is a service cell or service individual

that directly meets the given request. In case there isn't such a SC or SI, the request mediator creates a new composition request *CompReq* and calls *handleCompositionRequest(CompReq)* on the composition mediator [#3].

U-Node	Mediators	Initial Service Cells	Device Capabilities
U-Node1	- All mediators	- createMap(Location):Map - showPictureMarker(Map, Location, Picture)	
U-Node2	- SC Execution	- getLocation(): Location	- GPS
U-Node3	- SC Execution - Security - Context	- takePicture(): Picture	- Camera

Table 14 U-Nodes of the example scenario

The composition mediator creates its initial context from the given composition request and tries to search new services depending from the current composition context during each composition step [#3.*]. Therefore, the composition mediator creates a new discovery request *DiscReq* during every step of the algorithms and calls *handleDiscoveryReq(DiscReq)* on the discovery mediator [#3.*.1]. The discovery mediator in turn searches for matching services on the local U-node by subscribing the current discovery request *DiscReq* on the network interface [#3.*.2]. Other reachable nodes can publish matching services in the network [#3.*.3]. In this use case, the discovery mediator finds the four service cells depicted in Figure 6. The service cells *getLocation()* and *takePicture()* can only be executed on U-Node2 and U-Node-3, respectively. Therefore, these nodes publish the service interfaces to indicate that the service can be executed remotely only. In the next step, the request mediator calls *handleBindingRequest(AbstractPlan)* [#4] on the binding mediator to create the executable plan *ExecutablePlan* depicted in Figure 6 from the abstract plan created by the composition mediator.

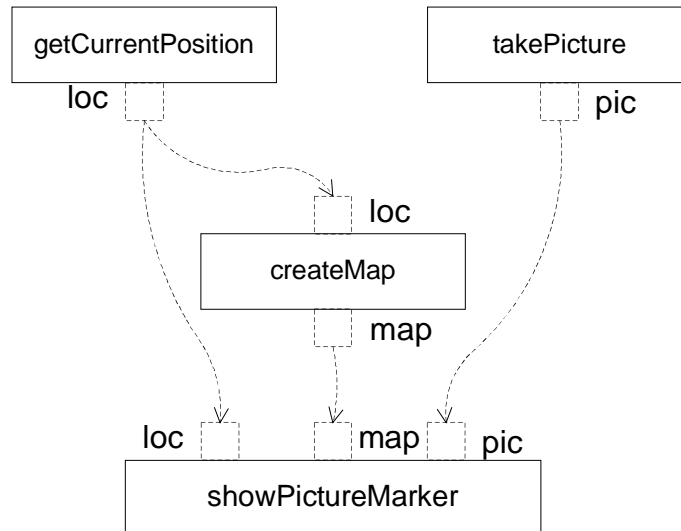


Figure 6: Executable plan (Service Individual)

Afterwards, the security mediator analyses the executable composition and adds security constraints by calling *addSecurityModifications(ExecutablePlan)* [#5] on the security mediator. In this example, if the services *getCurrentPosition* or *takePicture* require authorization, the Security Mediator extends the existing plan with an *Authorization* service added before these services. In the last step [#6], the request mediator calls *executeSI(ExecutablePlan)* on the SI execution mediator to run the executable SI. The SI

execution mediator calls internally *executeSC* on the SC execution mediator to execute the single service cells [#6.*]. Here, the SC execution mediator delegates the execution requests automatically to the associated node in case of remote execution. In this use case, *getCurrentPosition()* will be executed on the U-node2 and *takePicture()* on U-node3.

2.4.3 Conclusion

The Section 2.4 specified the mandatory set of mediators that are required to enable services to follow the BIONETS service life-cycle as defined in deliverable D3.2.1 [BIONETS_D321]. As outlined in Section 2.1, a custom set of mediators can be deployed on every node, where a required functionality that cannot be provided by local mediator can be proxied by another node hosting the respective mediator.

The mediator specification of this section serves as direct input for all implementation in work package WP5.

3. Co-operation with SP2 and WP4

3.1 Overview of Service Security Framework

Based on the previous sections in which we outlined the basic service framework functionalities, this section is going to review the basic security functionalities for BIONETS services. We will only outline the basic architectural features and their functionalities as Deliverable D4.6 [BIONETS_D46] will discuss the overall BIONETS security framework in more details, including the mechanisms used for securing the service framework.

3.1.1 Challenges

The challenges for the security framework with focus on BIONETS services were mainly investigated in Deliverables D3.1.1 [BIONETS_D311] and D4.2 [BIONETS_D42]. Summarizing these deliverables, the main challenges for securing services is the decentralized and disconnected nature of BIONETS which requires vivid interaction between heterogeneous devices which may run different types of services to accomplish a common task. These devices may additionally be associated with various users whom we don't know in advance and which cannot provide appropriate verifiable authentication tokens. Finally, to fit the autonomic characteristics of the service framework the security mechanisms deployed should be as transparent as possible to the user and the required user interaction should be minimal.

Thus, the security framework had to address the following aspects:

- Service identity management,
- Trust and reputation management, and
- Service configuration and execution management

to support a sufficient security level during the service life-cycle. This generally prevents services from

- Performing malicious actions without being detected, and
- Violating security policies of users or devices.

We will shortly outline the mechanisms developed to address the single aspects and explain how they can provide the service life-cycle with the listed security features.

3.1.2 Identity Management

For many services, identity management is essential in BIONETS. Services may run on behalf of users providing them with the expected functionality by distributing their required components in the network or collecting information, i.e. they have to provide appropriate credentials to perform a certain task or to consume certain types of data. Services are also subject to evaluation frameworks such as the trust and reputation management. Additionally, services may be used in business scenarios where sufficient identification and authentication is essential. Finally, services may use or offer security functionalities for which their integrity and authenticity has to be ensured.

Deliverable D4.5 [BIONETS_D45] defines the general identity management framework for the whole project including the management for services.

At the time when a service individual s is first generated, it is assigned a pair of public and secret keys, (P_s, S_s) and the service name $(ID_s, h(P_s), v_s, n_s)$. ID_s is basically the semantic description of s , v_s is the service version number, n_s is an optional, hierarchically structured, description of a service individual for distributed implementations. P_s is the underlying public key, which in practice is kept in the form of a hash value $h(P_s)$. v_s is used to mark significant changes in a service, which itself is expected to change frequently in BIONETS, due to evolution and adaptation. Thus, $(ID_s, h(P_s))$ identifies the service in a globally unique way and

should not change during the life cycle of the service, despite the service changes during its life cycle. In this way the identification management guards the origin and identity of the service and at the same time accounts for changes due to evolution and different distributions of the service in the network. The authentication of services is achieved by using digital signatures over the service name (which obviously includes the service identifier), its code-elements, and the signature meta-data such as the public key to be used for verifying the signature.

Of course, the use of a public key cryptographic system had to address the disconnected nature of BIONETS. Thus, schemes which allow the generation and verification of signature chains were introduced to ensure that some compromised key pairs do not compromise the whole system. Here, identity management system and trust and reputation system show a strong interdependence which appears to be natural for disconnected environments.

3.1.3 Trust and Reputation Management

The trust and reputation system for BIONETS services monitors behaviour using the feedback of the user or a particular metrics and tries to identify potentially malicious services. In this case, security is considered as a service quality, i.e. comparable to the biologically inspired monitoring and evaluation process described in the last section, the trust and reputation system evaluates and rates the trustworthiness of the service. This implies the assessment of whether a service respects certain contracts, e.g. whether it sends content it promised to send, whether it keeps content confidential or not, whether it forwards content to another service, etc.

The trust and reputation management which is defined, and elaborated in several deliverables [BIONETS_D41, BIONETS_D42, BIONETS_D45, BIONETS_D46] was designed in a bottom up fashion starting at the node level and culminating at the service level. This is due to the fact that the management system on the service level also depends on the management system on the node level. We will briefly describe the preliminaries to understand the relevance of the trust and reputation management system for BIONETS services.

As described above the structure of a service individual is hierarchical. Its structure can be described by the hierarchical field n_s in the service name. If a service individual is a service cell, i.e. there is no composition but a single cell which accomplishes a certain task, the trust and reputation system for services operates almost identically to the case where nodes are evaluated. Associating reputation with each service individual in a non-trivial service composition is a more complicated task. However, it enables a node to choose between service individuals with different but semantically equivalent compositions, where the overall reputation value of the service individual is either known or recursively computed on the basis of its components. Thus, the service framework is capable of autonomically selecting services which show a higher probability of appropriately executing a required task. This is done as follows: After the execution of s a node is required to formulate an opinion about the quality of the executed composition. But we also need to evaluate the behaviour of the lower levels of the service composition hierarchy which is organized in a tree-like structure. In general, we assume that the behaviour of some component can be evaluated (and in particular we can always evaluate the outcome of the root individual s), while others provide results whose quality is not measurable by itself. This implies that if the evaluation of the root service individual is low we are not able to isolate particular individuals or cells of this composition and identify faulty or malicious components.

To still provide some trust and reputation mechanisms we propagate the evaluation down the composition according to the following scheme:

- First, we try to assign an evaluation to each node in the composition tree.
- If some nodes remain unlabeled because it is not possible evaluate their contribution, we navigate the tree breadth-first
- If at least one sibling of node i is labelled with a reputation value lower than its parent, then i remains unlabeled.

- Otherwise (all siblings of i are either unlabeled or have a reputation higher than their parent), we label i with the parent's label (none if the parent was unlabeled).

The main drawback of this approach is the low reputation that can be achieved by an otherwise good individual or cell just by being often associated with a service which has low reputation. This problem can be overcome by frequently recombining components with the help of a randomization scheme in the service binding phase. Before executing a service, once all component reputations have been computed, node n will employ the following heuristic: If all components corresponding to a specific functionality within the service are below a given threshold, then one is chosen at random, with probability inversely proportional to the corresponding average information quality. So, unless a good candidate appears for a given function, all components will be used and will have a chance of being evaluated. For more details we refer to Deliverables 4.5 and 4.6 [BIONETS_D45, BIONETS_D46].

3.1.4 Secure Service Configuration and Execution Management

The trust and reputation management system presented above gives information about the past behaviour of a service. This information can influence future decisions on the use or preference of a particular service. However, security may define characteristics which may be violated as soon as a service starts running, e.g. highly confidential data may be published, an unauthorized process may turn the system into states which are reserved for authorized processes only, etc. Thus, processes which may conflict with installed security policies need appropriate assistance. Ideally, BIONETS should operate unattended. Thus, the assistance for services should operate autonomically.

For this purpose the service security framework implements *static* and *dynamic security checks* which ensure that the *service configuration* and *service execution*, respectively, comply with the security policies of the system.

This system is based on three main assumptions:

- users can specify security policies for their data,
- the security framework on a node can only enforce security policies for data items if they remain on this node, and
- an autonomic service security framework requires security service cells

In the first two assumptions the highly dynamic, decentralised, self-adapting nature of BIONETS and its demand for an autonomous operation are reflected. The first characteristic requires that all services which propagate through the system and which may implement new behaviour – generated by adaptation or evolution – should be executed. On the other hand, the system should only ask for minimum interaction with its users. Thus, adaptations or reconfigurations of services in order to make them compliant with specific security policies should be prevented. Additionally, the number of security checks, such as authentication tasks, should be minimized.

The static security checks in BIONETS ensure that service compositions are reconfigured in such a way that the information flow of a service does not conflict with the security policies attached to the data the service processes. This is accomplished by statically analysing a service composition prior to its execution. In case, the security characteristics of a data flow within a composition conflicts with the data security policies, the security framework tries to enhance the service composition with security service cells, e.g. authentication or authorization services, encryption or decryption services, etc. These security service cells reflect the third assumption which guarantees the required flexibility of the security framework.

Of course, not all security characteristics of the data flow may be known during composition time. As an example consider the security policy of a data item which specifies time, date, and the location. This context is only known during runtime. Thus, we also implement dynamic checks which verify these contextual data.

These mechanisms are described in more detail in the Deliverables [BIONETS_D42, BIONETS_D44, BIONETS_D46].

3.1.5 Conclusion

This section briefly presented the security mechanisms implemented in the BIONETS service life-cycle. The sketch of the security challenges inherent to BIONETS services together with the fundamental identity management form the basis for the brief presentation of two complementary security mechanisms: the trust and reputation and the secure service configuration and execution management. Finally, we outlined why the combination and integration of these two complementary mechanisms address the most common threats in BIONETS and form therefore important building blocks for the secure operation of autonomic services in BIONETS.

3.2 Algebraic analysis of service composition automata

3.2.1 Overview of algebraic and Lie group analysis of metabolic pathways

Over the past few years, and across the BIONETS and OPAALS projects, we have mobilised a growing number of researchers and institutions in order to deepen and extend our study of the formal structure of cellular processes as the principal source of inspiration in our approach to biologically-inspired computing. The rationale for this approach derives from the observation that the cell performs extremely complex functions by executing an intricate tangle of intersecting and interdependent metabolic and regulatory biochemical pathways, each of which can be considered analogous to an algorithm. Such behaviour is not planned and emerges spontaneously from the more elementary physical and chemical interactions at lower scales of description. What makes it possible is an immensely complex causal chain that links a stable periodic table of the elements to biological system behaviour through a stable set of physical laws. Thus, physical laws have bootstrapped an ecosystem that interacts with the biological systems living in it in such a way as to support their existence and reproduction.

Whereas the global and pervasive order construction mechanism that has enabled the bootstrapping of the ecosystem to happen is ultimately explainable through the memory-dependent dynamics of Darwinian evolution *acting across many generations*, a process called phylogeny, the same evolutionary mechanism has ‘discovered’ ways in which biological systems and their sub-systems can interact, internally and with their environments, in order to construct order *during the life of the individual*. These interactions are able to construct recursively nested complex structures at multiple scales, a process called ontogeny or morphogenesis, and then keep the metabolism of the adult organism running throughout its lifetime.

Rather than attempting to reproduce or emulate this overwhelmingly complex and physical order construction process, our approach has been to focus on the *output* of this process, in the form, for example, of a stable metabolic or regulatory pathway. The key concept is stability: in spite of the unpredictability of the biochemical inputs to the cell, appropriate cellular functions will be performed reliably, as long as the cell or the individual is healthy. As we discussed more extensively in D2.2.9 [BIONETS_D229], the concept of regularity (over space, time, or different scales of description) is best formalised through algebraic symmetries. Thus, our research programme has been to analyse the hierarchical algebraic structure of the mathematical models derived from metabolic pathways, to interpret what their dynamic and computational function might be, and to then apply these algebraic structures as constraints on automata in order to obtain a model of computation that we are calling Interaction Computing.

The evidence so far suggests that a stable pathway is not stable of its own accord, but is kept within the analogue of a stable ‘potential well’ by the pathways it is biochemically coupled to, which therefore act as constraints. As a consequence, it appears that in order to achieve the Interaction Computing vision (that we have so far developed only conceptually) we will have to understand how multiple threads, that are performing different algorithms, need to be coupled so that they can aid or constrain each other, as the case may be. In D2.2.9 we have

referred to this concept as the kernel of Symbiotic Computing, a model of computing that is meant to apply to the interaction of higher-level software constructs, such as whole services.

The predominant mathematical model used to analyse the interdependence of biochemical pathways is a set of coupled, and generally non-linear, ordinary differential equations (ODEs) derived from the chemical reaction equations. The set of dependent variables in such a set of ODEs is made up of the concentrations of compounds participating in the chemical reactions. Starting from these same chemical reaction equations, the system dynamics can be discretised as a Petri net, from which a finite-state automaton can be derived. The justification for this discretisation process has been discussed in D2.2.9. Our work, as reported in the present deliverable, has then focused on looking for algebraic structures in both mathematical models of the *same* pathway: Lie group dynamical symmetries in the system of ODEs and discrete computational symmetries in the automata.

We have focused on the p53-mdm2 regulatory pathway for the above analytical activities. Whereas the details of the Lie group analysis of this system are discussed in the Appendix of this report, the algebraic analysis of the semigroup associated with the corresponding automaton is being performed in the OPAALS project, so that we will only report the current results and insights here. P53 is a protein that participates in most regulatory pathways of the cell, and mdm2 is another protein that regulates the concentration level of p53. Depending on the p53 level, the cell can suspend its cycle to allow its DNA to be repaired, it can reach the state of reproductive senescence (completing its cycle but not reproducing), or it can immediately destroy itself. The malfunction of the p53-mdm2 system is believed to be related to approximately 50% of all cancers. It is therefore an interesting case to study in the development of a theory of autonomic computing, and for this reason we have been analysing it.

As discussed in Appendix C, the p53-mdm2 regulatory system [DBLML09] is characterized by oscillatory and non-oscillatory regimes. A simplification of the p53-mdm2 system we have been working with assumes that in the absence of DNA damage the response of the system to non-equilibrium starting values of p53 and mdm2 is to create a peak of p53-mdm2 until enough p53 is destroyed and its level is brought back to equilibrium, without oscillations. On the other hand, if p53* is present because of DNA damage, then the system responds with a damped oscillation in its p53 level, until the damage is fixed. This behaviour is confirmed by the mathematical analysis based on Lie groups that is discussed in Appendix E. There are two ways in which Lie groups can help us with this problem: (1) by helping us solve the system of equations, as reported in Appendix E in e.g. deriving a single first-order Riccati equation from the original system of 4 ODEs; or (2) providing additional information on the symmetry structure of the problem. This second aspect has not been pursued yet due to lack of time.

Although in the long term we believe that greater insight will be gained by understanding the relationship of these Lie symmetries to the observed dynamical behaviour, in the meantime we have developed an intuitive understanding of the relationship between the algorithmic symmetries of several biological pathways and the algebraic structure of the automata derived from them. In particular, cyclic or periodic behaviour in the physical/biological system appears to correspond to the presence of permutation groups embedded in the semigroups associated with the automata derived from these systems. Therefore, we expect to see more permutation groups in the model that corresponds to the oscillatory regime of the p53-mdm2 system than in the model that corresponds to the “healthy” regime. Figure 7 shows the automaton corresponding to a 2-level Petri net derived from the biochemical equations of the p53-mdm2 system [ENS09].

Although the deeper questions about the connections between the symmetries of non-linear dynamical systems and the symmetries of the finite-state automata derived from the same biochemical system will need to wait for more research before they can (hopefully) be answered, in the meantime, we can focus on the significance of the permutation groups for the nature of the algorithms they correspond to. This is what we will discuss in the rest of this section. In the remainder of this section we discuss how the automata concepts discussed above relate to the timed automata being used in the BIONETS Service Framework. We

present the latest ideas concerning a possible model of Interaction Computing. And we finally draw some conclusions on the architecture of algorithms for Interaction Computing.

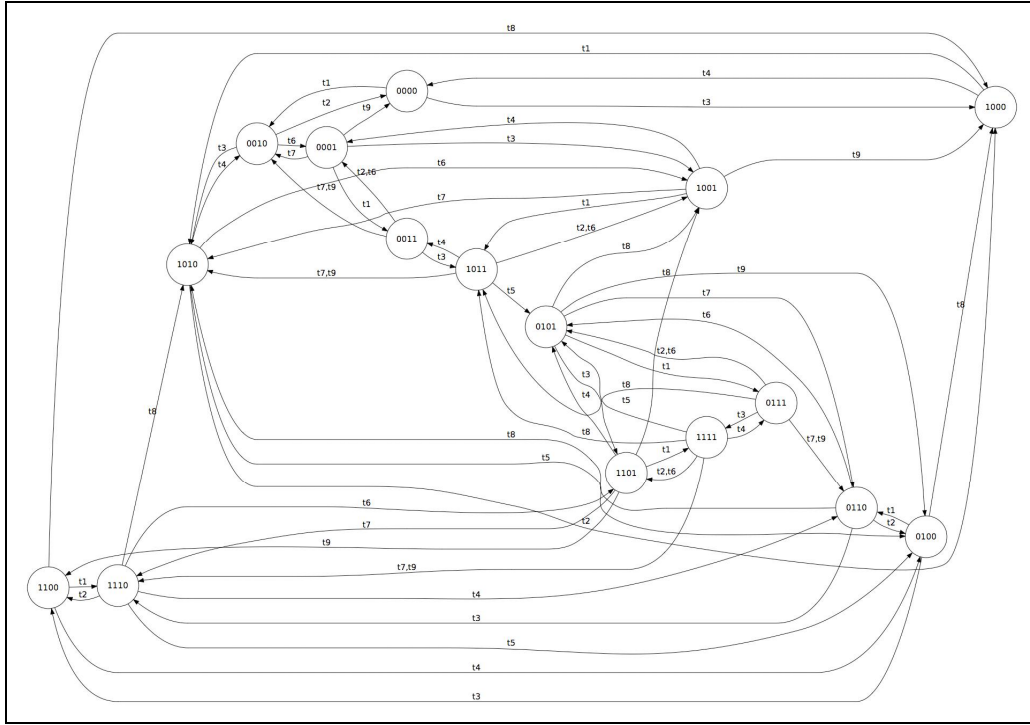


Figure 7: Automaton derived from 2-level Petri net of the p53 system ($2^4 = 16$ states)

3.2.2 Timed automata vs. regular automata

Timed automata and regular automata differ in two ways: whereas regular automata rely on inputs to trigger transitions, timed automata use the current state as an input; and, whereas regular automata do not have constraints on state transitions, timed automata have clock constraints (“invariants”) and constraints on system variables acting as inputs to states (“guards”), both of which, together, act as constraints on state transitions. The brief discussion below assumes the reader is familiar with deliverable D3.2.3 [BIONETS_D323], where the timed automata were originally introduced, and with the more recent article by Pfeffer [P09]. Timed automata do not operate on discrete time values, but on so-called clock regions, i.e. every property is verified or falsified with regard to clocks being in a specific range. Clock regions always hold boundaries for certain constraints, such as “x is true for all values of the clock c between 3 and 5”. In that way, model checkers operate on intervals of possible clock values and can therefore make statements on all values within this interval (without requiring discrete values here). Given the example of an invariant $c \leq 5$ and guard $c \geq 3$, a model checker would not evaluate every discrete step between 3 and 5, but would pass the transition and “remember” that this passage was only possible for $3 \leq c \leq 5$; all subsequent evaluations take this fact into account.

The constraints (i.e. time constraints or invariants and guards) can be seen as a super-structure that is necessary to ensure that the automaton does what it is supposed to do under the right conditions. In other words, the constraints are very useful for enforcing arbitrary modifications of behaviour externally and in a top-down manner. They are a very convenient engineering tool to stop a particular automaton from doing what it is not supposed to do and to force it to do what it is supposed to do. However, the introduction of constraints is not necessarily needed. More precisely, the constraints do not necessarily interfere with the natural progression of the service execution. In other words, it is possible to imagine a very special case in which the guards and the invariants, for a given TA, are *always* satisfied, so

that the various actions of the automaton are performed without any delays or interference caused by the constraints. In such a hypothetical (and admittedly improbable) case the timed automaton would behave in a way that is indistinguishable from the same base automaton without any time constraints or guards at all.

3.2.3 Conceptual description of a model of Interaction Computing

The concept of guard is applicable also to biochemical pathways. For example, it is quite common for a pathway to progress to a certain stage (we model reaching the completion of a particular biochemical reaction as reaching a particular state of the corresponding automaton), but then to be unable to progress to the next reaction until a particular substrate finds its way to the area of the cytoplasm where the pathway is ‘executing’. The need to wait for such a hypothetical substrate is entirely analogous to the imposition of a guard on a state transition in the form of a required system variable acting as an input to that state.

The above observation suggests that guards and time constraints (invariants) could indeed be used to model biological behaviour, but only after their role within the greater dynamical and computational framework of the pathway in question has been understood. In other words, because the pathways that are coupled to a particular pathway *A* are not only *limiting*, but also *enabling* the behaviour of *A*, the introduction of guards and constraints on *A* must reproduce the same effects in a precise computational sense. It follows that the dynamics of the broader system must be understood before the behaviour of *A* at its centre can be modelled correctly.

With the above in mind, the challenge of bio-computing that we have to solve is how to design an automaton, or a family of coupled automata, in such a way that the couplings create the necessary constraints on the ‘other’ automata, such that each of the automata behaves in the correct way to make the overall system do what it is supposed to, with a minimum number of constraints imposed from the outside.

Many years after proving the prime decomposition theorem for semigroups and machines, John Rhodes published a book that he had started working on in the 1960s and that has come to be known as the “Wild Book” [RN09]. In this book he provides a very clear definition of an alternative to a Turing machine, which we believe to be a very promising starting point for a model of interaction computing.

As we know, an algorithm implementable with a Turing machine is equivalent to the evaluation of a mathematical function. As Wegner and co-workers argued in a series of papers over the last 20 years [GW05 and references therein], the evaluation of a mathematical function can afford to take place by following an “internal clock”, i.e. the Turing machine is isolated from its environment while it evaluates the algorithm. Biological systems, on the other hand, are continually interrupted by external inputs and perturbations. As an example of this class of computations Golding and Wegner used “driving home from work”, which they described as a non-Turing-computable problem [GW05]. Turing himself had foreseen this possibility in his original 1936 paper as the “choice machine” [T36], although he did not pursue it further. Similarly, Rhodes defines a machine as a special mathematical function. Unlike a regular function from a set of inputs to a set of outputs, Rhodes’s function or sequential machine accepts an input at each discrete point in time and generates an output based on its state and on all the previous inputs up to that point. The realisation of such a machine is achieved through a finite-state automaton that Rhodes calls a “circuit”, $C(f)$. He goes on to say

The reason why Turing machine programs to realize a computable f are not unique and the circuit which realizes the (sequential) machine f (namely $C(f)$) is unique is not hard to fathom. In the sequential machine model we are given much more information. It is ‘on-line’ computing; we are told what is to happen at each unit of time. The Turing machine program is ‘off-line’ computing; it just has to get the correct answer – there is not time restraint, no space restraint, etc. [RN09: 58]

To develop this discussion further, it is helpful to provide a few familiar definitions, following Rhodes’s notation.

Definition 3.1. Let A be a non - empty set. Then $A^+ = \{(a_1, \dots, a_n) : n \geq 1, a_j \in A\}$.

A *sequential machine* is a function $f : A^+ \rightarrow B$, where A is the basic input set, B is the basic output set, and $f(a_1, \dots, a_n) = b_n$ is the output at time n if a_j is the input at time j for $1 \leq j \leq n$.

Thus, A^+ is just the free semigroup from basic automata theory, i.e. the set of all possible strings composable from a basic input alphabet using the concatenation operation. We are going to use a generalisation of the sequential machine, also by Rhodes, which produces an output for each input it receives and not just in correspondence of the most recent input. We are going to call this generalisation an *interacting machine*.

Definition 3.2. Let $f : A^+ \rightarrow B$ be a sequential machine. Then an *interacting machine* $f^+ : A^+ \rightarrow B^+$ is defined by

$$f^+(a_1, \dots, a_n) = (f(a_1), f(a_1, a_2), \dots, f(a_1, \dots, a_n)).$$

Such a machine will still accept inputs at each time step, but in contrast to the previous definition it will also produce outputs at each time step. The combination of two or more interacting machines will form an Interaction Machine (IM).

Going from the abstract definition of a function to a circuit that realises it is not, however, necessarily trivial. Although more details of how such a realisation might be constructed will be given in D4.6, we need to introduce here a few more facts and definitions in order to complete the conceptual model of the IM. Thus, we define a (Mealy-like) automaton, or ‘circuit’:

Definition 3.3. $C = (A, B, Q, \lambda, \delta)$ is an automaton with input alphabet A , output alphabet B , state set Q , next - state function λ , and output function δ iff A and B are finite non - empty sets, Q is a non - empty set, $\lambda : Q \times A \rightarrow Q$, and $\delta : Q \times A \rightarrow B$.

Rhodes goes on to prove that, given any f , a C can always be found, and proves that such a C is minimal, calling it $C(f)$. The main point remaining is that, if the possibility that Q be infinite is left open, as the above definition does, “then the output function $\delta : Q \times A \rightarrow B$ could be a badly non-computable function, with all the interesting things taking place in the output map δ , and we are back to recursive function theory” ([RN09]: 59). Therefore, we note the interesting conclusion that Q must be finite and that the realisation of an IM must be made up of interacting *finite-state* automata.

In the Wild Book, after completing the presentation of the algebraic theory of automata that he partly founded, Rhodes analyses the Krebs cycle in great biochemical and mathematical detail, and on the basis of this discussion proposes an abstract model of the cell, as shown in Figure 8. This model can be regarded as a more detailed instantiation of Rosen’s M-R system [BIONETS_D224] in a computer science context, but developed from a different mathematical viewpoint and firmly anchored in cell biochemistry and algebraic automata theory. For this reason we regard this model as the end-point of the conceptual development of the IM development.

From this sound mathematical basis, we feel that an automaton realisation of the IM can be attempted, with the objective of reproducing biological behaviour in computer science applications such as security and service composition. For example, the pathways shown in Figure 8 could be related to the BIONETS Service Individuals as a design pattern.

A significant amount of mathematics is still to be done in order to derive automata structure from desired behaviour, and the first steps in this direction will be reported in D4.6 and in the BIONETS eBook. In the meantime, having reached a greater clarity on the architecture of the IM, we are now in a position to begin the mapping of this structure to classes of possible behaviours that are compatible with it using category theory, in order to develop a behaviour specification language that can be transformed into biologically-inspired structures that realise it. This approach is discussed in the next sub-section.

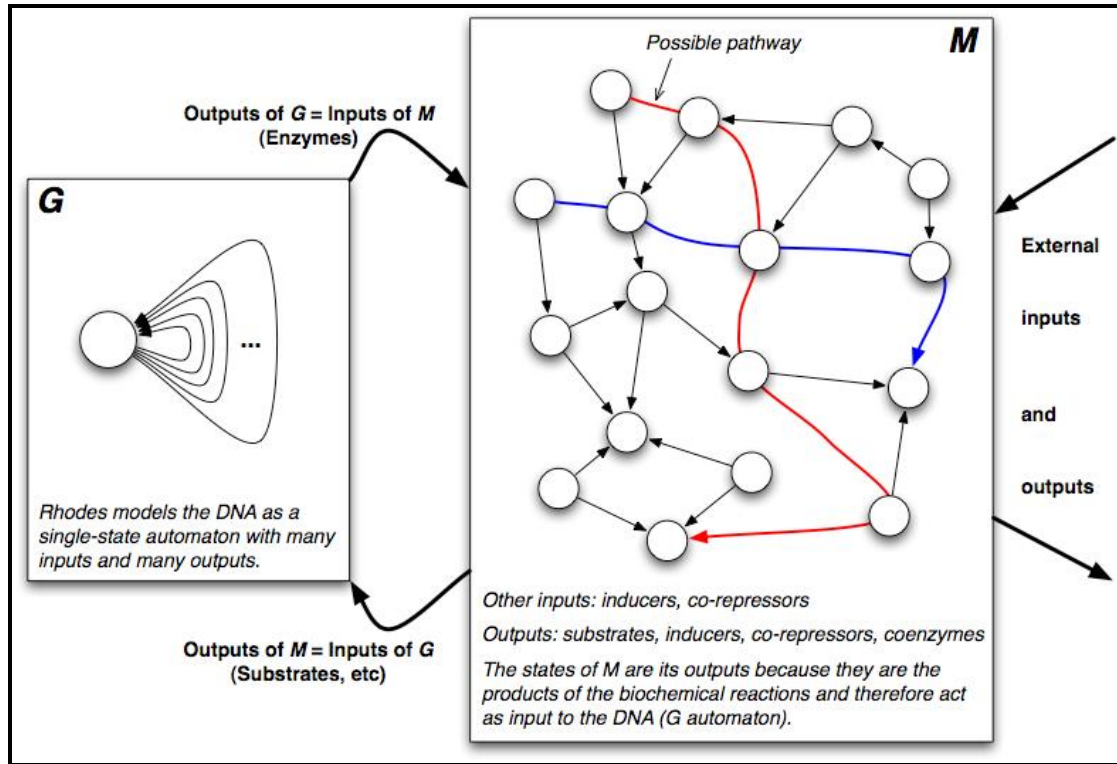


Figure 8: Conceptual architecture of the biological realisation of the Interaction Machine, based on Rhodes ([RN09]: 198).

3.2.4 Categories from Service Compositions

Deliverables D2.2.4 [BIONETS_D224] and D2.2.9 [BIONETS_D229] form the basis for this section. They studied category theory by introducing the most important concepts and sketched how category theory can be used to establish the important links between languages, automata, logic, semantics, and computational models in general. This is important as we consider category theory as the glue between the various models we have at hand and which we need in order to establish a unified framework of interaction computing. We will briefly outline how regular sequential systems, such as the automata presented above, their more sophisticated version, i.e. timed automata as introduced in Section 2.6.2 and used by modelling services in BIONETS, and non-linear systems, represented by differential equations, can be generalized by means of category theory. This is essential for developing the framework of interaction computing. In future steps we will be able to investigate the computational model defined by biological systems and derive an appropriate language able to describe the respective processes. To show the relevance of category theory and explain to a computer scientist how it becomes a powerful translation machine, we outline which relationships have to be exploited in order to transform between different instances of automata described in different computational models.

For this purpose, we first define an abstract transition system. Although its definition is very similar to the notion of automata introduced above, we do not call them automata because we want to avoid any confusion with the class of regular automata. A transition system is defined by a tuple (Σ, S, s_0, δ) , where Σ is the alphabet, S is the set of states, s_0 denotes the initial states, and δ is the transition relation with $\delta \subseteq S \times \Sigma \times S$.

With this general definition we can describe any automaton defined above in some monolithic form. However, if we consider a service composition in BIONETS we will realize that it consists of service individuals or service cells which interact with each other, concurrently, to generate complex services. Thus, modelling such a system also requires an abstraction into various transition systems which are able to interact. We take a similar approach as Winskel

and Nielsen [WN97] who modelled concurrent processes with an eye on the calculus for communicating systems (CCS) developed by Milner [Mil80]. From a BIONETS point of view, we can say that the way a service individual is communicating with another individual determines the relationship between them. For example, one individual may alter the execution flow by removing or adding certain branches in the flow. This can be seen as a type of atomic modification of the transition system. As a consequence, we could model such a communication between two services, or more precisely, between two transition systems T_1 and T_2 , as a transformation (σ, λ) where σ denotes a function from the states of T_1 to those of system T_2 where s_0 remains identical, i.e. $\sigma(s_0) = s_0$. λ is a partial function from the alphabet of T_1 to the alphabet of T_2 such that any transition in T_1 (s, a, s') is transformed into a transition $(\sigma(s), \lambda(a), \sigma(s'))$ of T_2 if $\lambda(a)$ is defined. If it is not defined then a kind of idle transition is defined for which $\sigma(s) = \sigma(s')$ holds. This allows the removal of certain transitions.

Finally, the transformation (σ, λ) forms a category \mathbf{T} in which the transition systems are the objects and (σ, λ) are morphisms on these objects. As a consequence, \mathbf{T} also supports important universal constructions, such as limits and co-limits, products, etc. (see Deliverables D2.2.4 [BIONETS_D224] and D2.2.9 [BIONETS_D229] for more details). This again induces other important operations such as restrictions and re-labelling. Combinations of the latter and of universal properties can help to define parallel composition and nondeterministic sums. The definition of these operations directly induces a process language able to describe the parallel, potentially nondeterministic, execution of the service compositions described in BIONETS. In our case, this is not extremely interesting because we already know several languages which could have been used to describe such compositions. Thus, we only sketch this derivation process to show the potential of category theory and the status at which we arrived. In fact, if we derive a description from a biological system such as the p53 system (see above), we will ultimately be able to define an execution model which reflects the particularities of the class of algorithms we are describing. In particular, we will need to find out how the universal properties in this category are defined and this in turn will help us to understand and define the required operations, such as composition and parallel execution. This basically results in a process category.

The concepts presented above were basically introduced in 1993. Their elaboration and extension include the definition of *interaction categories* by Abramsky [Abr96]. Abramsky chose a new type of substitution and thereby introduced the notion of processes. Thus, in interaction categories, objects are basically interpreted as interface specifications and processes are maps between these specifications. Abramsky additionally unified various methodologies, starting at non-well-founded sets – the origin of category theory – ending at concurrency theory, and defined the interaction categories SProc and ASProc for synchronous and asynchronous processes. Cockett and Spooner showed that SProc and ASProc can be constructed by using systems of open maps [RCS94] and can also be embedded in the category of processes. Finally, Worytkiewicz, followed yet another approach [Wor02]. He slightly modified the substitutions above by considering a transition system as a type of graph which attaches semantic information to the edges of the graph, which basically represents a control flow graph. Worytkiewicz calls this system categorical transition system because the semantic information appears to be best organized in a category.

Why do we summarize these different attempts, here? In particular, we want to highlight that there is not one category which possibly arises from one particular automaton. As we have seen – SProc and ASProc are just two examples – the contrary is the case. The different aspects of the underlying computation systems have to be investigated and their characteristics are reflected in the appropriate category. Thus, the category we will derive from biological systems in the future may also fit in the class of process categories. So, the appropriate representation of these systems in a specification language will need more human expertise. The tools used for all the insights listed above are based on the concepts of bi-

simulation, open maps, adjunctions and monads, and Kleisli- as well as Eilenberg-Moore categories. This list does not only partially reflect our research agenda but also shows the links between categorical structures important for our research. Why is this?

Open maps can be interpreted as so-called path-lifting properties [JNW94]. Here, paths represent computations, e.g. sequences of consecutive transitions or a partial order of events, respectively, depending on what kind of representation has been chosen. Assume a morphism $f : X \rightarrow Y$ which can transform these paths into other paths while preserving behaviour. f is called an open map if it preserves the labels on the path, and if a path can be extended in Y using f then this extension can be matched to an appropriate extension in X . Obviously, this model can be considered as a general definition of bi-simulation which tries to relate states of a transition system which behave identically and are not distinguishable by an external observer. Thus, in our context open maps can be used as a more concrete definition for the general notion of bi-simulation applied to different types or categories of transition systems.

Adjunctions are in particular relevant for our research as they on the one hand allow for the linkage between an automaton and its behaviour (see also Deliverable D2.2.9), and on the other hand they enforce a particular structure on the category they start from. This special structure is called a monad. In fact, every adjunction gives rise to a monad on the category the adjunction starts from. Monads are an important concept in formal languages as they are able to model effects [Wad92] such as global state manipulation, exception handling, text parsing, iterations, invoke continuations, simple express types of sequential computation. It is helpful to draw an analogy between adjunctions and regular functions. Adjunctions can be seen as a pair of functors where one functor between two categories is the inverse of the other; thus, the conditions they impose on the starting category are in some way analogous to the conditions a regular function needs to satisfy in order to be invertible.

Finally, the Kleisli- and Eilenberg-Moore categories are very special categories as they are initial and terminal objects respectively of the categories of adjunctions. This is particularly interesting as due to these characteristics the two categories can be used to answer the question: Given a monad C , can we construct an adjunction which gives rise to this monad? With these ingredients we can! In fact, if we review the above and extend our understanding of category theory and the categorical characteristics of the automata derived from biological systems we may derive for certain monads which are linked to biological system the corresponding adjunction which again gives rise to the category which finally describes the behaviour of the corresponding machine. In this way, category theory may pave the way for a better understanding of biological systems.

Assuming the understanding of this “behaviour category” and its influence on the monad it was basically derived from, i.e. its actual implementation, we imagine to use another functor which translates this behaviour category into a specification category which can be used to build a specification language for the Interaction Machine described above. Additionally and as expressed in Deliverable D2.2.9, the concept of minimal realisation which uses adjunctions to derive a realisation from a specific behaviour – in our case – from a behaviour specification of a system, can be used to actually guide the automatic generation of a system which actually implements this behaviour.

3.2.5 Conclusion

In this section we have given an update of our work on the development of a biologically-inspired interaction computing model and of its formalisation through a corresponding specification language. Although the work is not finished the intermediate results and insights we have obtained are encouraging and motivate us to keep developing these ideas. Some of the steps following these ideas will be discussed in D4.6, in the BIONETS eBook, and in the OPAALS and other follow-on projects.

4. Assessments

The purpose of this document was to conclude the research work and concepts of sub project 3 for topics related to BIONETS autonomic service life-cycle and to provide final outcome of SP3 with regard to a novel life-cycle support that enables services to evolve over time to remain responsive to environmental changes.

In this document we have concluded the work done in the BIONETS SP3. We have reported the current state of our research in several topics researched in the WP3.2. We have concluded what is the BIONETS service and how it achieves its autonomicity. We have reviewed the role played by biologically-inspired aspects in the definition and refinements of the BIONETS service framework. We have provided an overview of service framework business models. We have also presented achievements of co-operation with SP2 and WP4 by defining an algebraic analysis of service composition automata and providing the basic security functionalities for BIONETS services. Finally, we provided a definition of all the mediators in the BIONETS service framework needed to achieve autonomic service life-cycle.

The purpose of SP3 was to design and implement an innovative framework for the creation, dissemination and evolution of autonomic services, able to survive and evolve in a resource constrained, dynamic and heterogeneous environment, without relying on a centralized control. In this assessments section we will summarize the most essential research challenges and their solutions developed in SP3 to achieve this goal and also results from co-operation with SP2 and WP4 related to service framework issues.

The SP3 was divided to three work packages 3.1, 3.2 and 3.3. In the WP3.1 the main result was the developed innovative SerWorks architecture [BIONETS_D311, BIONETS_D312, BIONETS_D113/313, BIONETS_D114/314]. The architecture incorporates service and network architectures and benefits from the biologically inspired communication paradigms. The WP3.2 deepened the research by developing innovative approaches for support of the whole service life-cycle [BIONETS_D321-6]. In the WP3.3 economic viability of the developed system was studied, from the point of view of all potential actors and the potential impact of the envisioned solutions from a user centric point of view [BIONETS_D332, BIONETS_D334].

The BIONETS service life-cycle can be roughly divided to three main phases; *Preparation and Creation*, *Evolution* and *Integration*. The *Preparation and Creation* phase comprises activities around acquisition of user needs and environment context as well as the creation of functionally appropriate Services addressing. The *Evolution* phase describes all activities concerning improvement, adaptation and proactive creation of Services. Finally yet importantly, the activities of *Integration* phase address the actual handling of Services in BIONETS environment. [BIONETS_D114/314]

The service autonomicity in BIONETS is achieved mainly through different phases of service life-cycle, controlled by service mediators. These mediators, presented and defined, in the Section 2.4, provide the implementations of different solutions developed in the SP3.

The mediators encapsulate management and control functionality to support the BIONETS service life-cycle. Thus, mediators provide functionality that supports a certain degree of autonomy, self-awareness, and self-control of services. As the deliverables in the scope of SP3 of the BIONETS project have been subjected to specific aspects of the service life-cycle introduced in the first deliverable of this sub-project, the mediators can be considered as the final outcome of SP3 with regard to a novel life-cycle support that enables services to evolve over time to remain responsive to environmental changes.

In the Table 15 below we collect the most essential research results achieved both in SP3 and by co-operation with SP2 and WP4 for achieving autonomic service life-cycle. These results are divided according to the main phases of service life-cycle.

<i>Preparation and Creation</i>	<i>Evolution</i>	<i>Integration</i>
Self-healing distributed services: a solution, based on the use of techniques inspired by artificial embryogenies, was developed to provide automatic recovery following nodes' failures in a distributed service settings. Numerical simulations were carried out for a distributed service monitoring use case [MLY09].	Service evolution at the parameter level: A solution, based on a cooperative genetic algorithm, was developed to enable self-configuration and self-optimisation of service parameters. The developed solution was shown to converge to the optimal operating point by means of numerical simulations [TMCDP09].	Service deployment / Service penetration: A solution for autonomic service migration of was developed enabling the autonomic migration of Service Cells between U-Nodes. The developed solution was verified with simulations and prototype implementation. [LRHK09]
Establishment of the socio-economic foundations for potential BIONETS business models. [BIONETS_D332, BIONETS_D334]	Integration of Service Architecture with socio-economic foundations for development of BEBS. [BIONETS_D332, BIONETS_D334]	A simulation model for the illustration of the economic models was developed BIONETS Economic and Business Model [BIONETS_D332]
Definition and specification of a timed automaton based workflow model to represent service individuals. The model allows for the dynamic integration of services and supports the evolution of SIs [BIONETS_D321][PLS08].	Definition and implementation of evolution of service individual level through the application of genetic operators [LPS08]. Simulation results proved the feasibility of the approach [BIONETS_D341]	A platform integration workflows and components, allowing automatic creation and evolution of complex services [submitted to IJARAS]
Definition and implementation of an evolutionary algorithm for the effect-driven automatic creation of service individuals [BIONETS D326].	The algorithm for the automatic creation of SIs can also be applied to already existing SIs during runtime such that composed services can be evolved at runtime to remain responsive to environmental changes [BIONETS_ D326].	Security mechanisms reconfiguring the service composition and controlling the service execution in such a way that conflicts with the security policies set by a user are avoided are implemented in the prototype [CBLP09]
Specification of ADL language for creation of computers (Functional + Non-functional architecture) [Autonomics 07, BIONETS_D3.2.2, BIONETS_D3.2.6]	Specification and implementation of a fuzzy logic based model to represent non-functional service properties that serve as a basis to evaluate service cells and service individuals with regard to given preferences. Fitness evaluation and service transformation (achieved through the application of genetic operators to SIs) thus paves the way towards evolutionary services.[PKS08,KPS09,BIONETS_D325]	
Inclusion of security mechanisms into service life-cycle to support the secure reconfiguration of a service during the preparation as well as creation phase [BIONETS_D113/D313]	Specification and implementation of a language for distributed reconfiguration of component systems [SINTER09]	

Table 15: SP3 Research Challenges and Solutions with Autonomic Service Life-cycle

5. References

- [**ABR96**] S. Abramsky, S. Gay, R. Nagarajan, Interaction Categories and the Foundations of Typed Concurrent Programming in Proceedings of the NATO Advanced Study Institute on Deductive program design, pp. 35-113, Springer-Verlag New York, Inc., 1996.
- [**AGA04**] J. Agar, "Constant Touch: A Global History of the Mobile Phone". London, UK, Icons Book Ltd. 2008.
- [**ADMS09**] E. Altman, P. Dini, D. Miorandi and D. Schreckling (Eds.), "Paradigms for Biologically-Inspired Autonomic Networks and Services", BIONETS Project eBook, Feb. 2009.
- [**AL91**] R. B. J. T. Allenby, Rings, Fields and Groups: An Introduction to Abstract Algebra, 2nd Edition, Oxford: Butterworth, 1991.
- [**BAR02**] A. L. Barabasi, "Linked: The New Science of Networks". Cambridge USA, Perseus Publishing Books. 2002.
- [**BG98**] V. Bala, and S. Goyal "Learning from neighbours." Review of Economic Studies 65(3): 595-621. 1998.
- [**BEN05**] Y. Benkler "Sharing nicely" The Yale Law Journal 114: 273-358. 2005.
- [**BEN06**] Y Benkler "The Wealth of Networks: How Social Production Transforms Markets and Freedom". New Haven and London, Yale University Press. 2006.
- [**BHN09**] F. Baude, L. Henrio, P. Naoumenko, "Structural reconfiguration: an autonomic strategy for GCM components", in Proc. of IEEE ICAS, 2009.
- [**BIONETS_AS07**] BIONETS Multiple authors Application Scenarios – Round II. Internal Document. 2007.
- [**BIONETS_D111**] D. Miorandi (Ed.), "Application scenario analysis, network architecture requirements and high-level specification", BIONETS D1.1.1, Jul. 2006.
- [**BIONETS_D112**] D. Miorandi (Ed.), "Architecture, Scenarios, and Requirements refinements", BIONETS D1.1.2, Aug. 2007.
- [**BIONETS_D113/313**] D. Miorandi (Ed.), "D1.1.3/3.1.3 SerWorks architecture v1.0," BIONETS Deliverable (D1.1.3/3.1.3), Jul. 2008.
- [**BIONETS_D114/314**] J. Huusko (Ed.), "D1.1.3/3.1.3 SerWorks architecture v2.0," BIONETS Deliverable (D1.1.3/3.1.3), Jul. 2008.
- [**BIONETS_D122**] D. Raz and F. De Pellegrini (Eds.), "Disappearing Network Autonomic Operation and Evolution", BIONETS D1.2.2, Jul. 2007.
- [**BIONETS_D228**] L. Yamamoto (Ed.), "Robustness and Evolvability of Artificial Polymer Chemistries in Online Evolution, BIONETS Project Deliverable D2.2.8, Nov. 2009.
- [**BIONETS_D311**] J. Lahti (Ed.) "Service architecture: requirement specification and concept definition", BIONETS D3.1.2. Jul. 2007.
- [**BIONETS_D312**] J. Huusko (Ed.) "Refinement of Service Architecture and Requirements", BIONETS D3.1.2. Jul. 2007.
- [**BIONETS_D321**] H. Pfeffer, D. Linner, S. Steglich and I. Radusch (Eds.), "Specification of Service Life-Cycle", BIONETS D3.2.1, Feb. 2008.
- [**BIONETS_D322**] L. Yamamoto (Ed.), "D3.2.2 Specification of Service Evolution", BIONETS D3.2.2. Jul. 2007.
- [**BIONETS_D224**] P. Dini, (Ed.), "D2.2.4: Evolution and Gene Expression in BIONETS: A Mathematical and Experimental Framework". <http://www.bionets.eu>: BIONETS Deliverable, European Commission, 2008.
- [**BIONETS_D229**] P. Dini, (Ed.), "D2.2.9: Mathematical Framework for Interaction Computing with Applications to Security and Service Choreography". <http://www.bionets.eu>: BIONETS Deliverable, European Commission, 2009.
- [**BIONETS_D323**] F. Baude and L. Henrio (Eds.), "Graph-based Service Individual specification: Creation and Representation", BIONETS D3.2.3, Jan. 2008.
- [**BIONETS_D324**] J. Lahti (Ed.), "Advanced Service Life-Cycle and Integration", BIONETS D3.2.4, Jul. 08.

- [**BIONETS_D325**] H. Pfeffer (Ed.), "Evaluating the Fitness of Service Compositions", BIONETS D3.2.5, Dec. 2008.
- [**BIONETS_D326**] F. Baude and L. Henrio (Eds.), "Bio inspired service creation and evolution", BIONETS Project Deliverable D3.2.6.
- [**BIONETS_D332**] BIONETS Multiple authors ID3.3.2 Economics for BIONETS Business Models. 2007
- [**BIONETS_D41**] R. Cascella and Anurag Gard (Ed.) "Trust and reputation management system definition", BIONETS D4.1. Jun. 2007.
- [**BIONETS_D42**] D. Schreckling (Ed.) "Towards security in BIONETS", BIONETS D4.2. Aug. 2007.
- [**BIONETS_D44**] D. Schreckling, "Adaptive Security in BIONETS", BIONETS D4.4. Feb. 2009.
- [**BIONETS_D45**] M. Brunato (Ed.) "Reputation, Authentication, Anonymity, and Privacy in BIONETS", BIONETS D4.5. Sep. 2009.
- [**BIONETS_D46**] D. Schreckling (Ed.) "Security in BIONETS", BIONETS D4.6. Dec. 2009.
- [**BLH10**] F. Baude, V. Legrand, L. Henrio, P. Naoumenko, H. Pfeffer, L. Bassbouss, D. Linner, "Mixing Workflows and Components to Support Evolving Services", to appear in International Journal of Adaptive, Resilient and Autonomic Systems, 2010.
- [**BIONETS_D54**] I. Carreras (Ed), "Pervasive ubiquitous peer-to-peer context-aware application", BIONETS D5.4, Aug. 08.
- [**BS08**] V. Baka, and S. V. Scott "From Studying Communities To Focusing On Temporary Collectives: Research-In-Progress on Web 2.0 in the Travel Sector". Information Systems Working Papers. London, Department of Management, London School of Economics and Political Science: 12. 2008.
- [**CAM99**] P. Cameron, Permutation Groups, Cambridge University Press, 1999.
- [**CAM08**] P. Cameron, An Introduction to Algebra, 2nd Edition, Oxford University Press, 2008.
- [**CBLP+09**] I. Carreras, L. Bassbouss, D. Linner, H. Pfeffer, V. Simon, E. Varga, D. Schreckling, J. Huusko, H. Rivas, "BIONETS: Demonstrating Self Evolving Services in Opportunistic Networking Environments", in Proceedings of BIONETICS 2009, Avignon, France, December 9-11, 2009.
- [**CDPMW05**] I. Carreras, F. De Pellegrini, D. Miorandi and H. Woesner, "Service Evolution in a Nomadic Wireless Environment", in Proc. of WAC, Athens, Oct. 05.
- [**D04**] P. Dittrich, Chemical Computing, in: Unconventional Programming, Paradigms (UPP 2004), Springer LNCS 3566, 2005, pp. 19–32.
- [**DBLML09**] P. Dini, G. Briscoe, I. Van Leeuwen, A. J. Munro, and S. Lain, D1.3: Biological Design Patterns of Autopoietic Behaviour in Digital Ecosystems. http://files.opaals.org/OPAALS/Year_3_Deliverables/WP01/:OPAALS_Deliverable, European Commission, 2009.
- [**ES03**] A. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer, 2003.
- [**ENS09**] A. Egri-Nagy, C. L. Nehaniv, M. J. Schilstra. "Symmetry Groups in Biological Networks", Information Processing in Cells and Tissues, IPCAT09 Conference, 5-9 April 2009. Journal preprint.
- [**FIELDING00**] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.
- [**FRO07**] Frogloop "Is it Worth it? An ROI Calculator for Social Network Campaigns." Retrieved February, 2009, from <http://www.frogloop.com/care2blog/2007/7/17/is-it-worth-it-an-roi-calculator-for-social-network-campaign.html>. 2007.
- [**GH06**] J. Goldbeck, and J. Hendler "Inferring Trust Relationships in Web-Based Social Networks." ACM Transaction on Internet Technology 6(4). 2006
- [**GB09**] M. C. Gonzales, and A.-L. Babarasi "Complex networks - From data to Models." Nature Physics 3. 2009.
- [**GRA82**] Granovetter, M. The Strength of Weak Ties: A network theory revisited. Social Structure and Network Analysis. P. V. Marsden and N. Lin. Beverly Hills, USA, SAGE: 105-130. 1982.

- [GW05] D. Golding, and P. Wegner. "The Church-Turing Thesis: Breaking the Myth", Computability in Europe (CiE) conference series, 2005.
- [HAR08] R. Harwood, "Nesta Connect: Connecting Dots and Valuing Networks." Retrieved Jan 17, 2009, from <http://blogs.nesta.org.uk/connect/2008/12/connecting-dots-and-valuing-networks.html>. 2008
- [HKS08] H. Pfeffer and S. Krüssel and S. Steglich (2008). "Fuzzy Service Composition Evaluation in Distributed Environments". In Proceedings of The International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services (I-CENTRIC 2008), Sliema, Malta, October 2008, IEEE Computer Society, Conference Publishing Services (CPS), Los Alamitos, 335-342.
- [JEL05] M. Jelasity, Engineering emergence through gossip. Proceedings of the Joint Symposium on Socially-Inspired Computing Hatfield, UK, University of Hertfordshire. 2005
- [JNW94] A. Joyal, M. Nielsen, G. Winskel, "Bisimulation from Open Maps", Journal of Information and Computation, vol. 127, pages 164-185, 1994.
- [KC03] J. Kephart, D. Chess, The vision of autonomic computing, IEEE Computer, 36 (1) (2003) 41–50.
- [LPRS07] D. Linner, H. Pfeffer, I. Radusch, S. Steglich, "Biology as Inspiration Towards a Novel Service Life-Cycle", in Proc. of ATC 2007, pages 94-102.
- [KPS09] S. Krüssel, H. Pfeffer, and S. Steglich, "Fuzzy Modeling of Resource Consumption for Service Composition Evaluation". In Proceedings of the Second International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2009.
- [LPS07] David Linner, Heiko Pfeffer, and Stephan Steglich, "A genetic algorithm for the adaptation of service compositions", in Proc. of BIONETICS 2007, Budapest, Hungary.
- [LRHK09] J. Lahti, H. Rivas, J. Huusko, V. Könönen, "Simulation and Implementation of Autonomic Service Mobility Framework" BIONETICS 2009, December 9-11, 2009 - Avignon, France.
- [MAK09] Maki "The Value of Friends in Social Media Websites." Retrieved Jan 2009, 2009, from <http://www.doshdosh.com/the-value-of-friends-in-social-media-websites/>. 2009.
- [MLY09] D. Miorandi, D. Lowe and L. Yamamoto, "Embryonic Models for Self-Healing Distributed Services", to appear in Proc. of BIONETICS 2009, Avignon, Dec. 2009.
- [MIL80] A. R. G. Milner, Calculus of communicating systems, Springer LNCS, 1992
- [MYDP10] D. Miorandi, L. Yamamoto and F. De Pellegrini, "A Survey of Evolutionary and Embryogenic Approaches to Autonomic Networking", to appear in Computer Networks, 2010, doi:10.1016/j.comnet.2009.08.021
- [MSTY08] T. Meyer, D. Schreckling, C. Tschudin, L. Yamamoto, "Robustness to Code and Data Deletion in Autocatalytic Quines", Transactions on Computational Systems Biology X, LNBI 5410, 2008.
- [MY08] D. Miorandi and L. Yamamoto, "Evolutionary and Embryogenic Approaches to Autonomic Systems", in Proc. of Inter-Perf 2008.
- [MYBT09] T. Meyer, L. Yamamoto, W. Banzhaf, and C. Tschudin. Elongation Control in an Algorithmic Chemistry. In Proceedings of the 10th European Conference on Artificial Life (ECAL 2009), Sept. 2009.
- [NS04] T. Nakano, T. Suda, "Adaptive and Evolvable Network Services". In: Proc. Genetic and Evolutionary Computation Conference LNCS 3102 151–162 (2004)
- [NS05] T. Nakano, T. Suda, "Self-organizing network services with evolutionary adaptation". In: IEEE Trans. on Neural Networks (2005)
- [OASIS06] OASIS Reference Model for Service Oriented Architecture 1.0, Official OASIS Standard, Oct. 12, 2006 [url:<http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>].
- [PLRS07] H. Pfeffer, D. Linner, I. Radusch, and S. Steglich, "The bio-inspired Service Life-Cycle: An Overview", in Proc. of IEEE ICAS, 2007.
- [PLS07] D. Linner and H. Pfeffer and S. Steglich (2007). "A genetic algorithm for the adaptation of service compositions". Proceedings of Workshop on Technologies for Situated and Autonomic Communications (SAC) at 2nd International Conference on Bio-Inspired

Models of Network, Information, and Computing Systems (BIONETICS 2007), Budapest, Hungary, December 10-12th.

[PLS08] Heiko Pfeffer, David Linner, and Stephan Steglich, "Modeling and Controlling Dynamic Service Compositions", in Proc. of IEEE ICCGI, 2008.

[P09] Heiko Pfeffer, "An Underlay System for Enhancing Dynamicity within Web Mashups", International Journal on Advances in Software, Vol 2, No 1, 2009.

[PRI07] Priscilla "Does effort = effect?". Retrieved February, 2009, from <http://www.solidariti.com/article/Doesefforteffect/>. 2007

[QUI06] J. Quiggin, "Why do social networks work?". Retrieved January, 2009, from <http://crookedtimber.org/2006/05/60/why-do-social-networks-work>. 2006

[RCS94] J. Robin B. Cockett, David A. Spooner, SProc Categorically in Proceedings of 5th International Conference on Concurrency Theory, pp. 46-159, LNCS, Springer 1994

[RH02] Rheingold, H. Smart Mobs. Cambridge, USA, Perseus Publishing. 2002

[RN09] J. Rhodes, Applications of Automata Theory and Algebra, Via the Mathematical Theory of Complexity to Biology, Physics, Psychology, Philosophy, and Games, C. L. Nehaniv Ed., New Jersey: World Scientific, 2009.

[SCH09] Schrock, A. Examining social media usage: Technology clusters and social network site membership. First Monday 14 . 2009

[SM03] K. O. Stanley, R. Miikkulainen, A taxonomy for artificial embryogeny, Artif. Life 9 (2003) 93–130.

[T36] A. Turing. "On Computable Numbers, with an Application to the Entscheidungsproblem", Proc. London Math. Soc., 42:2, 1936, pp. 230-265; A correction, ibid, 43, 1937, pp. 544-546.

[TMCDP09] D. Tacconi, D. Miorandi, I. Carreras and F. De Pellegrini, "Cooperative Evolution of Services in Ubiquitous Computing Environments", submitted to ACM Trans. Aut. Adapt. Systems, 2009.

[TWI09] Twitter "Twitter Value." Retrieved February, 2009, from <http://tweetvalue.com>. 2009.

[VAN09] E. Van Buskirk, "5 Ways the Cellphone Will Change How You Listen to Music." Retrieved January 2009, from <http://blog.wired.com/business/2009/01/six-ways-cellph.html> 2009.

[WAD94] P. Wadler, "Comprehending Monads", Mathematical Structures in Computer Science, Special issue of selected papers from 6'th Conference on Lisp and Functional Programming, 2:461-493, 1992.

[WEI07] D. Weinberger, Everything is Miscellaneous - The Power of the New Digital Disorder. New York, Times Books. 2007.

[WIR06] Wiredset "Types of Engagement." Retrieved February, 2009, from http://wiredset.com/archives/2006/11/20_engagement.html .2006

[WN97] G. Winskel and M. Nielsen, Categories in Concurrency in Semantics and Logics of Computation, Cambridge University Press, pp. 299-354, 1997

[WOR02] K. Worytkiewicz, Concrete Process Categories in Electr. Notes Theor. Comput. Sci., vol. 68,1:70-85, 2002.

[Y10] L. Yamamoto. "Evaluation of a Catalytic Search Algorithm". submitted for publication.

[YSM07] Lidia Yamamoto, Daniel Schreckling, Thomas Meyer, "Self-Replicating and Self-Modifying Programs in Fraglets", in Proc. of BIONETICS 2007.

Appendix A: Terminology

- **T-Nodes.** T-Nodes are simple, inexpensive devices with sensing/identifying and basic communications capabilities. T-Nodes act as an interface with the environment and are used to gather contextual information which is utilized by the U-Nodes to provide context-awareness. T-Nodes do not communicate among themselves but are just read by U-Nodes in proximity. They present minimal requirements in terms of processing/storage/communications.
- **U-Nodes.** U-Nodes are complex, powerful electronic devices with computing and communication capabilities. PDAs, laptops and smart phones represent examples of a U-Node. U-Nodes are typically carried around by users and therefore are inherently mobile. Their mobility is exploited, in BIONETS, to provide system-wide diffusion of messages. U-Nodes host services. They interact with the environment through T-Nodes, from which they gather the contextual information necessary to provide the users with services enhanced by context-aware features. U-Nodes may communicate among themselves to exchange information, such as environmental data or service-specific code (in order to enable service evolution).
- **Access Points (APs).** Access Points are complex powerful devices that may be used for (i) accessing IP-based services by the BIONETS networks (ii) collecting environmental data (through BIONETS system) from a remote IP service (iii) providing IP shortcuts among disconnected BIONETS islands. APs are envisioned to act as *proxies* between BIONETS networks and IP networks.
- **Service cell (SC):** an atomic logic entity, which may provide knowledge, content, or functionality to other services and users.
- **Service individual (SI):** a SI is a composition of Service Cells. Service Individuals may also be composed of other Service Individuals in a recursive way. Service Individuals are expressed with a graph representation [BIONETS_D323], wherein nodes represent types of Services (the elements of the composition); edges (links) represent the interactions (communication) between the Services.
- **Mediators:** logic entities performing control functions related to the operations of SCs and SIs. In particular, mediators implement the autonomic functionalities encompassed by the BIONETS service lifecycle.
- **Messages.** Communications in BIONETS are based on the exchange of messages. Messages are service data unit, i.e., encapsulation of data items meaningful to a service. In general, messages will be much larger than standard IP packets. (This is because single IP packets usually do not expose meaningful data to the service layer.) Messages will consist of a payload (or content) and metadata (expressed as a set of <attribute,value> pairs) carrying the necessary information for the node to decide which operations should be undertaken. Communications in BIONETS are asynchronous and connectionless. Messages are treated as datagrams, and the whole system can be thought as a message-switching engine.
- **Opportunistic Forwarding:** mechanism for diffusing information in a highly partitioned network, based on the exploitation of “contact opportunities” between nodes in the system. Opportunistic forwarding is based on localized interactions only and exploits mobility of the nodes to ensure network-wide diffusion of messages.
- **Information Filtering:** mechanism for limiting the diffusion of data messages with low information content. This is related to the fact that, in most context-aware applications, context-related data loses its usefulness when being far (in both space and time dimensions) from the originating context. Information filtering is an essential building block of data management in BIONETS systems.
- **Identifier:** an identifier is a finite sequence of symbols of a given alphabet, used to identify an entity within a set of entities. Identifiers have a scope (in space and time) which determines the domain within which they can be used for identifying entities.

- **Name:** location-independent (i.e., with global spatial scope) identifier of a logical BIONETS entity. A name in BIONETS is constituted by a set of pairs < attribute, value >. Names in BIONETS are intentional, i.e., they can be used by services and applications to specify what they are looking for. Names in BIONETS are dynamic, i.e., they may change over time (equivalently: they have a limited scope in time).
- **Address:** location-dependent identifier (i.e., with local spatial scope) of a logical BIONETS entity (in this deliverable: node). Addresses can be used for identification purposes only within a connectivity island. Addresses have a limited scope in time. Addresses can be generated on-the-fly according to a random procedure by each U-Node and by some classes of T-Nodes. Addresses are numeric sequences and can be dynamically bound to names. Addresses may be used for performing one-hop point-to-point communications among BIONETS entities.
- **Identity:** globally unique identifier associated to each U-Node (and to some classes of T-Nodes). An identity has global scope in both time and space. Being location-independent, an identity is technically a name (even if we will use the two terms separately to avoid confusion).
- **Security principal:** any node in the BIONETS network architecture that can be authenticated.

Appendix B: Service Migration Mediator

The main component providing the service migration behavior is the Service Migration Mediator (SMM). The SMM is responsible for monitoring the service environment and triggering migration when needed. The SMM observes the overall state of the whole Node Island. It collects data from other mediators in the same node and also communicates with other Service Migration Mediators in the same Node Island for getting all the relevant information to make rational decisions.

The SMM migration models supported by SMM can be separated in three different types depending the trigger, movement model and environmental conditions:

1. *Node Island Migration:* This is the simplest case. The service is migrated from node A to node B inside the same node island. The motivation can be e.g. low-resource status in the originator node or that the hosting node is leaving the node island where the service has users.
2. *Locally Bound Migration:* In this case the service is not coupled so much with one Node Island or with its users, but some other motivation, e.g. hosting “hot-spot” –service in a limited geographical area. The basic difference with case 1 is that the triggers for migration does not come inside the node island, but from some “higher purpose” (e.g. load balancing).
3. *Long Distance Migration:* In this case the triggers for migration can be same as in both previous cases. The difference is the movement model used for migration. In the previous cases we make the migration from one node to another, one hop and knowing both end points. In this case we can make more than one hop and the destination node might not be the end point where the service stays to live. The originator node just initiates the migration and gives the service to some neighbouring node that then passes it to next one, and so on, until the suitable host node is found. With this mechanism we expand the service mobility outside the node island and can achieve better scalability and performance in certain scenarios.

Migration Decision Process

The services make their decisions at some particular time instances. The decision process can be initiated regularly or it can be triggered by some external signal, for instance the lack of battery power of the current platform. The Table 16 below presents different types of triggers utilized in the decision process.

Resource based	Service based	User based	Others
Node static resources (NSR)	HighServiceUsage (HSU)	HighDemandService (HDS)	ConnectionFaling Trigger (CFT)
Node dynamic resources (NDR)	LowServiceUsage (LSU)	LowQualityService (LQS)	HotSpot Trigger (HST)
Node network capacity (NET)			

Table 16: Migration triggers

At each time when a service makes its decision, it has a fixed set of options available corresponding to the above discussed actions, namely service migration, stopping the service or producing a new service. For each possible option, the utility value is computed by using the utility functions. An option with the highest utility value is then selected for execution.

Moreover, one can also connect a probability value for each option for expressing uncertainty associated with the option. The utility value is then weighted by the probability. The decision process is depicted below in Figure 9.

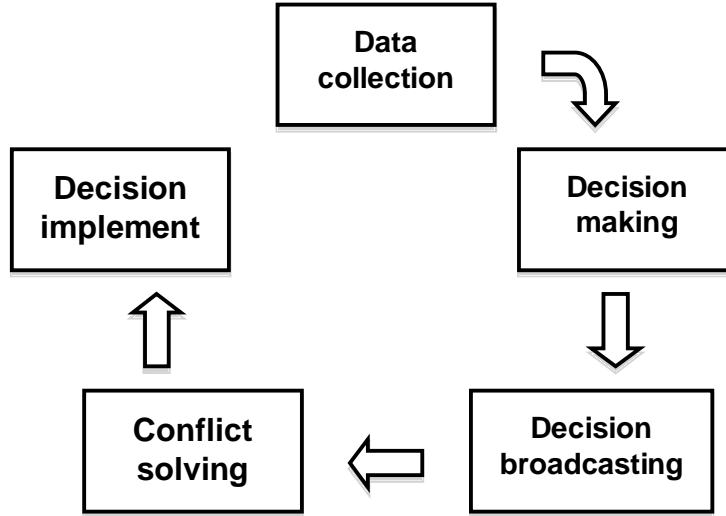


Figure 9: The steps of decision process.

The data collected in the collection phase can be related to service parameters such as request frequencies to the service, to the node hardware such as battery level or to network parameters such as link quality. SMM can also define reliability to the information it has, for example reflecting the age of the data.

SMM has as many migration options available as there exists nodes in the Node Island. For each migration option SMM computes a utility value that is based on the data it has available. SMM then broadcasts this decision to the other SMM's in the same Node Island. The broadcasting phase is required for possible conflict situations. For example, consider the case in which nodes A and B might both want to migrate services to a node C based on the current CPU load information available from C. However, if both A and B implement their decisions, the outcome might be highly unoptimal, the node C might not have enough resources for both services. There are several ways to solve conflict situations: for example conflicting SMM may select the decisions with the second highest utility values or only one, randomly selected, SMM is allowed to implement its decision.

The decision process and the used utility functions change depending whether the SMM is acting on a service provider or service consumer mode. With the service provider mode the SMM is monitoring the services running inside the host node. At each time when a SMM makes its decision it executes the functions for available actions for each service. For example, below is a simplified process for deciding what to do to a service S locating in the host node:

```

- run_utility_functions_for (S);
  -> choose highest from (f1, f2, f3, f4);
  f1 = stay_at_node
  f2 = migrateToX
  f3 = openMigration
  f4 = deprecation
- If (highest = f1): do_nothing();
- If (highest = f2): execute f2x to f2y (selectDestinationFromNodes (A, B, C...));
- If (highest = f3): execute f3x to f3y (selectDestinationFromNodes (A, B, C...));
- If (highest = f4): deprecate();
  
```

Figure 10 below present a sequence diagram for migration of Service X from Node A to Node B based on the resource warning trigger from NodeMediator.

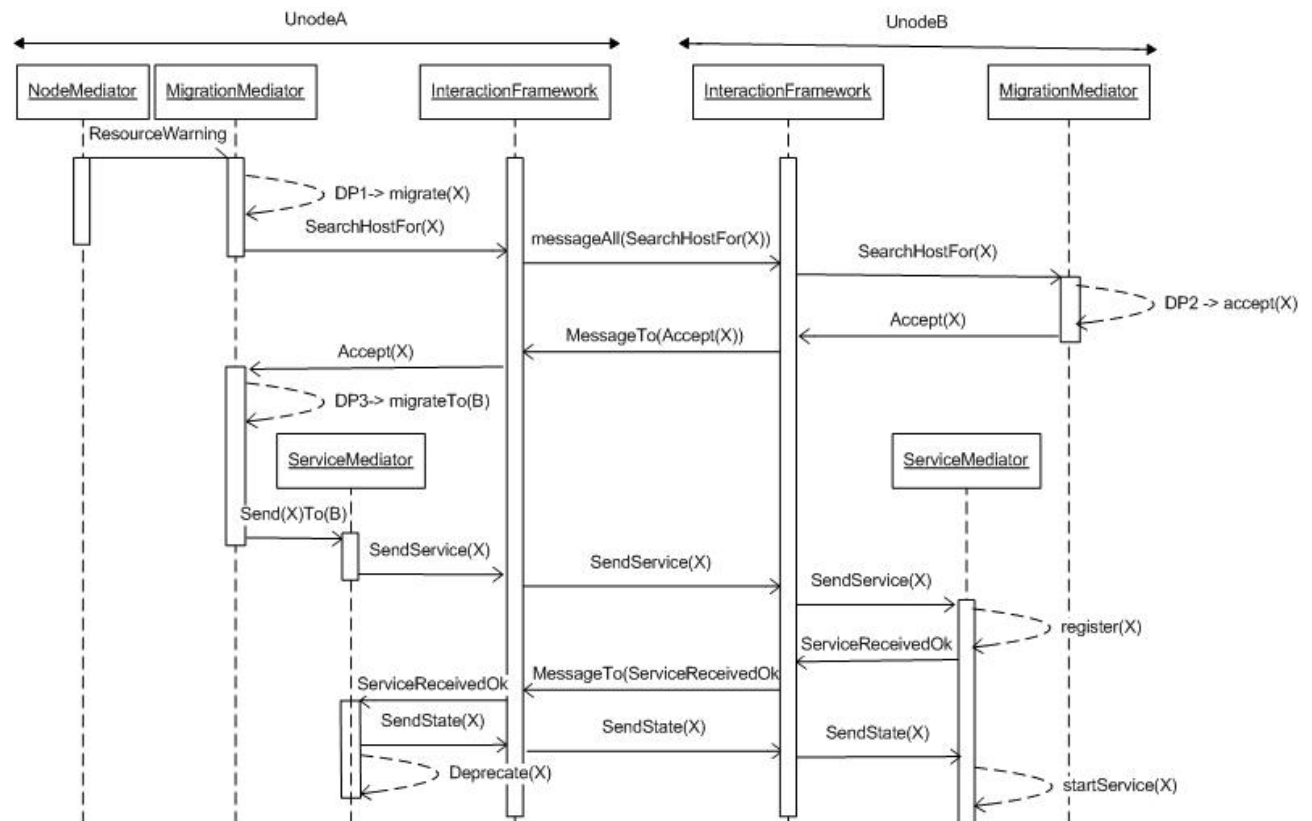


Figure 10: ServiceMigrationMediator Sequence Diagram

C Summary of Some Elementary Finite Group Facts

This appendix continues the presentation of finite group theory that was started in D2.2.4 and D2.2.9 [BIONETS_D224, BIONETS_D229]. The objective is to provide a conceptual toolkit for improving the understanding of algebraic structures, in support of the hypotheses advanced in the main text about what they might be for when they show up in ‘biological semigroups’.¹

C.1 Commutativity and non-abelian groups

Much of the richness of group theory arises from the fact that most groups are not abelian. In other words, the pairwise composition of their elements does not, in general, commute:

$$ab \neq ba, \quad a, b \in G \quad (1)$$

We employ the standard notation $H \leq G$ for “ H subgroup of G ”, counting on the context to clarify when this symbol means a subgroup relationship vs. a partial order. In Chapter 4 of D2.2.4 [BIONETS_D224] we talked about how the theory of group actions, orbits, and stabilisers can be used to unify different structures related to conjugation. We are revisiting these same concepts here, but using commutativity as the organising principle.

C.2 Group elements as internal maps

To build up the basic group concepts from the point of view of commutativity it helps to go back to the right regular representation and Cayley’s theorem that we discussed in D2.2.9, in which each group element $g \in G$ can be seen to operate on all the elements of G , including itself, through composition. In other words, the closure property which says that the composition of two elements $g, h \in G$ must equal another group element $k \in G$,

$$hg = k, \quad (2)$$

can also be seen as a map by right multiplication, i.e.

$$g : G \rightarrow G \quad (3)$$

and, in particular,

$$g(h) = k. \quad (4)$$

By the same token, the same element could be acting on any subset or subgroup of G , thereby mapping such a subset to another, not necessarily disjoint, subset. If however g is operating by right multiplication on a subgroup $H \leq G$ of which it is not a part, then it will generate a right coset, which *is* disjoint from H .

From this point of view, the generation of a right coset Hg can be seen as a transformation of the subgroup H . Carrying the point further, the operation of conjugation can also be seen as the inverse of this transformation, as shown in Figure A. For the sake of completeness, we are not referring to the inverse of the coset itself, which is

$$(Hg)^{-1} = g^{-1}H^{-1} = g^{-1}H. \quad (5)$$

Rather, as shown in the figure, if the generation of a coset is a transformation f ,

$$f(H) = Hg, \quad (6)$$

then we are calling conjugation the inverse of this function:

$$f^{-1}(Hg) = g^{-1}H. \quad (7)$$

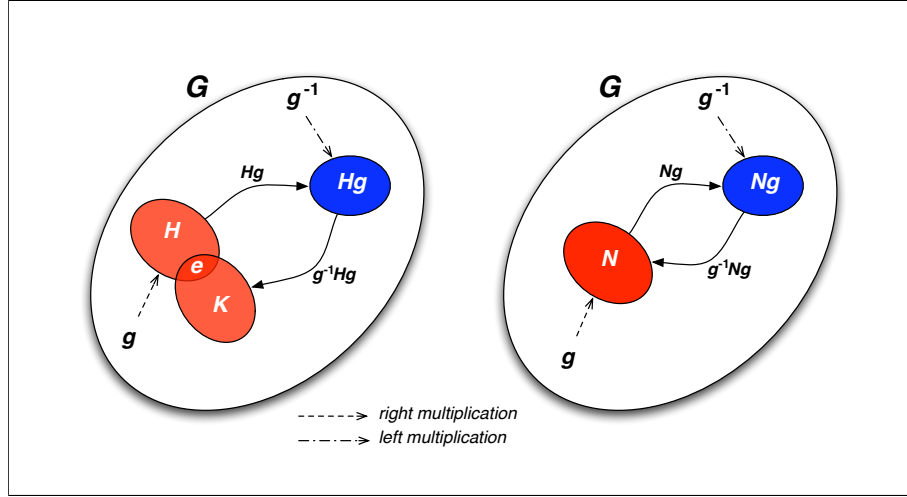


Figure A: Conjugation of a subgroup H and of a normal subgroup N

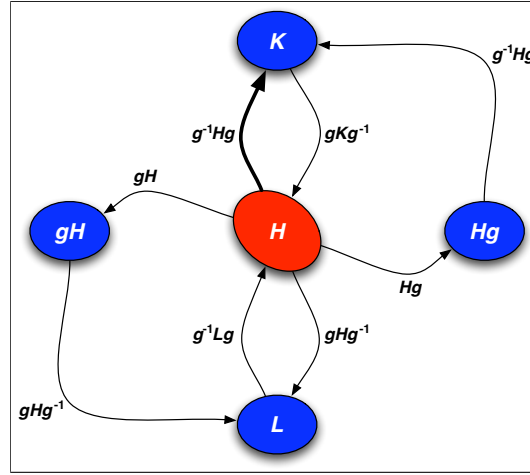


Figure B: Cosets and conjugates of non-normal subgroup H by element g ($K \cap H \cap L$ not shown)

From the figure we can see that there are in general two scenarios. When this inverse operation yields back the starting subgroup, N , we recognise N as a normal subgroup and we treat conjugation as a proper inverse of the coset operator. When, instead, the subgroup in question is not normal, H in the figure, conjugation will yield a different subgroup K , such that $e \leq H \cap K < H$.

If conjugation is seen as an operator that generates a subgroup from another subgroup, we can visualise and connect some of the concepts we have already encountered, as shown in Figure B for a non-normal subgroup H . In this figure, the thicker arrow indicates the normal interpretation of subgroup K as the conjugate of H under g . But we also show how the same conjugate subgroup can be seen as the ‘failed inverse’ of the coset operator. Further, since for a non-normal subgroup the right and left cosets are not equal, a symmetrical situation occurs ‘on the other side’.

Transformations from G to itself are of two kinds: those that preserve the structure of the group operations between the elements and those that do not. Those that do are isomorphisms, but because they map the group to itself they are called *automorphisms*. To test whether a

¹Semigroups associated with automata derived from metabolic pathways.

bijjective map θ from $(G, *)$ to (H, \cdot) is an isomorphism it is sufficient to test whether

$$(g_1 * g_2)\theta = g_1\theta \cdot g_2\theta \quad (8)$$

Thus, to test whether a map $\phi : G \rightarrow G$ is an automorphism one needs to test whether

$$(g_1 \cdot g_2)\phi = g_1\phi \cdot g_2\phi \quad (9)$$

Conjugation is an automorphism, right multiplication is not, and taking the inverse of an element is an anti-automorphism (because it inverts the order of multiplication).

C.3 Some important subgroup structures

There are some basic kinds of subgroup structures whose abelian properties can be used to present them in an integrated way that is easier to remember, as shown in Table A.

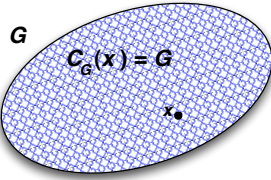
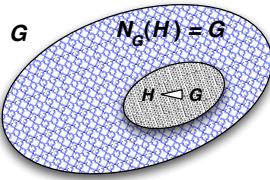
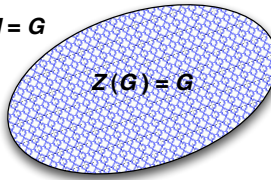
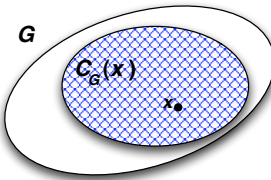
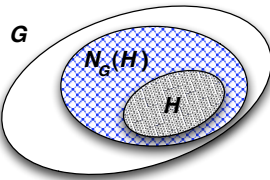
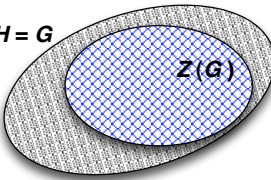
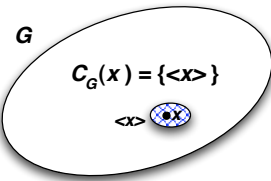
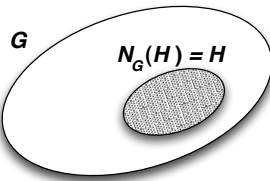
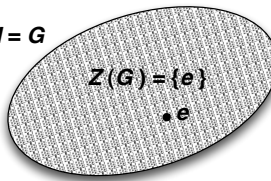
Largest possible centraliser (G abelian) 	Largest possible normaliser (Normal subgroup) $H \trianglelefteq G \iff gH = Hg, \forall g \in G$ 	Largest possible centre (G abelian) 
Smallest possible 'subgroup' (Centraliser) $C_G(x) = \{g \in G : xg = gx\}$ 	Normaliser $N_G(H) = \{g \in G : Hg = gH\}$ 	Largest possible subgroup (Centre) $Z(G) = \{x : gx = xg, \forall g \in G\}$ 
Smallest possible centraliser 	Smallest possible normaliser (G far from abelian) $N_G(H) = \{g \in H : Hg = gH\}$ 	Smallest possible centre 

Table A: Limiting cases of some important subgroup structures

This table shows how these different structures can be organised in two dimensions: in the vertical direction we show the degree of commutativity of different subgroup structures that are shown in increasing size in the horizontal direction: a single element, a proper subgroup, and a subgroup that is as large as the whole group.

D Background for p53-mdm2 system

A good discussion of the p53-mdm2 system is provided in [DBLML09], so here we summarise the highlights not already covered in the main text or in Appendix E.

We picked the p53-mdm2 regulatory pathway to work with because it is being investigated experimentally by the University of Dundee as part of their research in the OPAALS project. Although the p53 protein participates in many pathways in the cell, we focussed on a small subset as a starting point. Figure C shows a subset of the chemical reactions associated with this pathway.

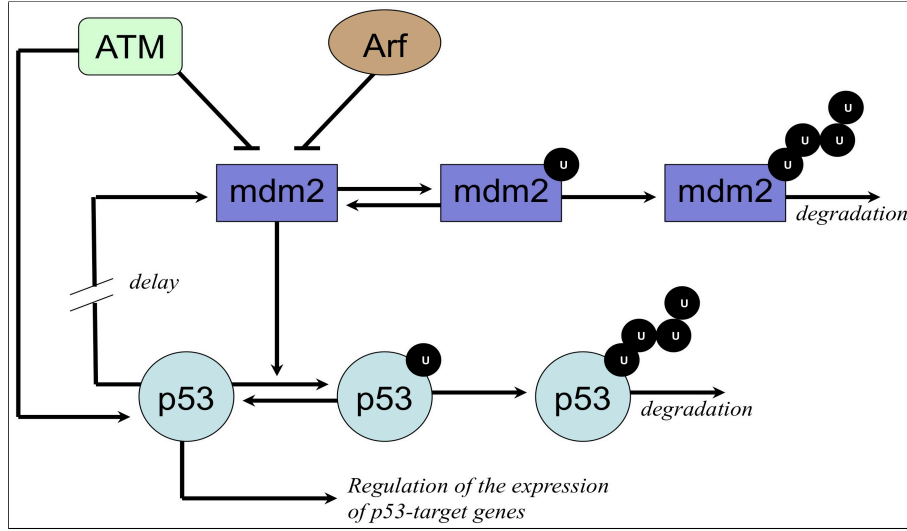


Figure C: Simplified interaction network for the p53-mdm2 regulatory pathway [DBLML09]

This system has been modelled by neglecting ATM and Arf for the moment, and by identifying different forms of the p53 and mdm2 proteins, as follows: $P_I = p53$, $M = mdm2$, $C = p53\text{-}mdm2$ compound, $P_A = \text{phosphorylated } p53$.

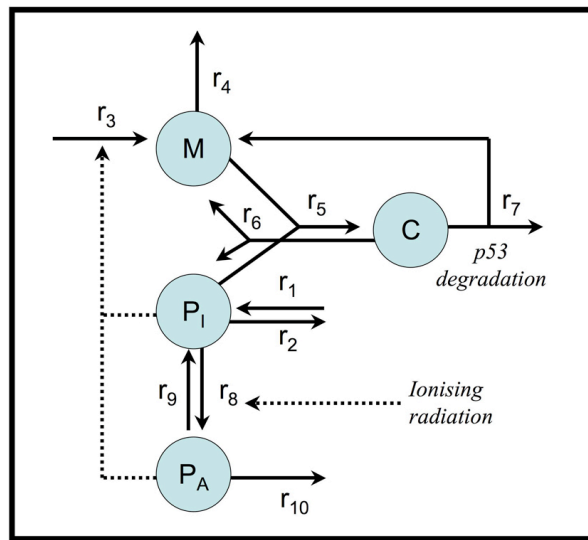


Figure D: Schematic of the simplified p53-mdm2 regulatory pathway [DBLML09]

The Petri net that corresponds to the simplest form of this pathway is shown in Figure E,

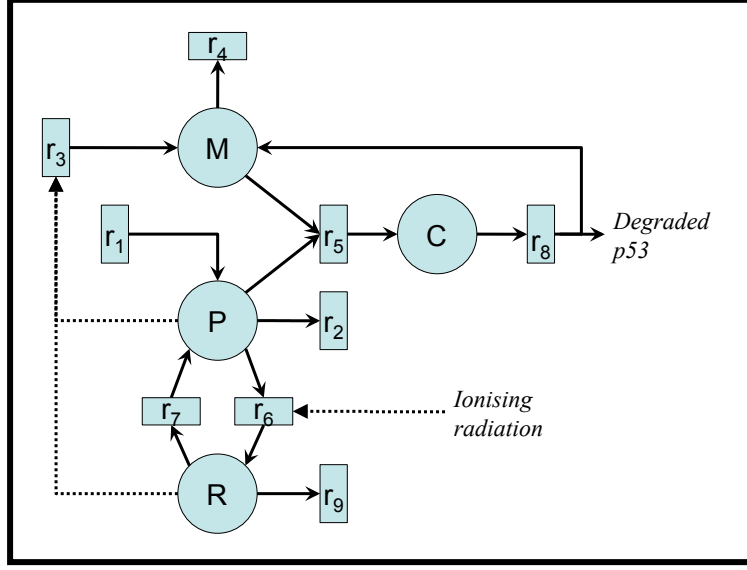


Figure E: Petri net for the simplified p53-mdm2 regulatory pathway [DBLML09]

where the symbols have been changed slightly: $P = P_I$ and $R = P_A$.

An automaton that corresponds to a 2-level Petri net is shown in Figure 7 of the main text. This is a very coarse discretisation since it corresponds to each place of the Petri net having either one token (concentration = 1) or zero tokens (concentration = 0). Because there are only two possible states for each place and there are 4 places, the automaton has 16 states in all. We have begun to analyse the 2-token case (3 levels, and therefore 81 states), but this automaton is already too large for brute-force Krohn-Rhodes decomposition and we are still looking for the most effective analysis strategy. As mentioned in the main text, this work is funded by the OPAALS project and therefore any results from this analysis will be reported in the final OPAALS deliverables, in 2010.

The next section of this appendix discusses the Lie group analysis of this same system.

E Lie Symmetry Analysis of p53-mdm2 System

E.1 The general symmetry conditions

We continue our work on the symmetry analysis on systems of differential equations started in Deliverable D2.2.9 [BIONETS.D229]. We use the methods described in D2.2.9 to understand the system of differential equations modelling the p53 network described in the OPAALS Deliverable D1.3 [DBLML09]. This model uses four variables, each depending on the time t . To understand why this is the case, we need to give a bit of background on this system.

The p53 protein is linked to many other proteins and processes in the cell, but its coupling to the mdm2 protein appears to be particularly important for understanding cancer. Depending on the concentration level of p53, the cell can (in order of increasing concentration): (1) operate normally; (2) stop all functions to allow DNA repair to take place; (3) induce reproductive senescence (disable cellular reproduction); and (4) induce apoptosis instantly (cell “suicide”). Therefore, p53 is a very powerful and potentially very dangerous chemical that we carry around in each of our cells, whose control must be tuned very finely indeed. Roughly 50% of all cancers result from the malfunction of the p53-mdm2 regulatory pathways in damaged cells that should have killed themselves. P53 levels are controlled by a fast feedback mechanism in the form of the mdm2 protein. P53 is synthesised all the time, at a fairly fast rate; but the presence of p53 induces the synthesis of mdm2, which binds to p53 and causes it to be disintegrated. When the DNA is damaged (for instance by radiation in radiotherapy) the cell responds by binding an ATP molecule to each p53, bringing it to a higher energy level that prevents its destruction and causes its concentration to rise. Thus there are in all 4 biochemical species: p53, mdm2, p53-mdm2, and p53*, whose concentrations are modelled by 4 coupled and non-linear ordinary differential equations (ODEs).

Therefore, we examine first a general system of four ODEs in four variables:

$$\begin{aligned} u_t &= f(t, u, v, y, z), \\ v_t &= g(t, u, v, y, z), \\ y_t &= p(t, u, v, y, z), \\ z_t &= q(t, u, v, y, z). \end{aligned} \tag{10}$$

As in D2.2.9, the subscript t denotes derivation with respect to time. In D2.2.9 we explained that a symmetry of a system of differential equations is a mapping which moves solution curves to solution curves. Every symmetry has an infinitesimal generator, which is a vector field \mathbf{v} of the form

$$\mathbf{v} = \tau(t, u, v, y, z) \partial_t + \phi(t, u, v, y, z) \partial_u + \psi(t, u, v, y, z) \partial_v + \mu(t, u, v, y, z) \partial_y + \nu(t, u, v, y, z) \partial_z. \tag{11}$$

Here, $\partial_t = \frac{\partial}{\partial t}$, $\partial_u = \frac{\partial}{\partial u}$, $\partial_v = \frac{\partial}{\partial v}$, $\partial_y = \frac{\partial}{\partial y}$, $\partial_z = \frac{\partial}{\partial z}$ denote the corresponding basis vectors, or equivalently the corresponding partial differential operators. If \mathbf{v} is an infinitesimal generator of a symmetry, then the flow of the symmetry (i.e. the mapping $(t, u, v, y, z) \rightarrow \Psi(\varepsilon, t, u, v, y, z)$) can be computed by exponentiating:

$$\exp(\varepsilon \mathbf{v})(t, u, v, y, z) = \Psi(\varepsilon, t, u, v, y, z), \tag{12}$$

where

$$\Psi(0, t, u, v, y, z) = (t, u, v, y, z), \tag{13}$$

$$\frac{d}{d\varepsilon} \Psi(\varepsilon, t, u, v, y, z) = \mathbf{v} \big|_{\Psi(\varepsilon, t, u, v, y, z)}. \tag{14}$$

Let \mathbf{v} be an infinitesimal generator of a symmetry of (10) in the form of (11). Moreover, for an arbitrary function $\zeta(t, u, v, y, z)$ let us define the following operator:

$$S(\zeta) = \zeta_t + f\zeta_u + g\zeta_v + p\zeta_y + q\zeta_z. \tag{15}$$

Then the symmetry conditions for the system (10) are the following:

$$-f_t\tau - f_u\phi - f_v\psi - f_y\mu - f_z\nu - fS(\tau) + S(\phi) = 0, \quad (16)$$

$$-g_t\tau - g_u\phi - g_v\psi - g_y\mu - g_z\nu - gS(\tau) + S(\psi) = 0, \quad (17)$$

$$-p_t\tau - p_u\phi - p_v\psi - p_y\mu - p_z\nu - pS(\tau) + S(\mu) = 0, \quad (18)$$

$$-q_t\tau - q_u\phi - q_v\psi - q_y\mu - q_z\nu - qS(\tau) + S(\nu) = 0. \quad (19)$$

Let us observe that for arbitrary $\tau(t, u, v, y, z)$ the following coefficients always satisfy the symmetry conditions (16–19):

$$\phi(t, u, v, y, z) = f(t, u, v, y, z) \cdot \tau(t, u, v, y, z), \quad (20)$$

$$\psi(t, u, v, y, z) = g(t, u, v, y, z) \cdot \tau(t, u, v, y, z), \quad (21)$$

$$\mu(t, u, v, y, z) = p(t, u, v, y, z) \cdot \tau(t, u, v, y, z), \quad (22)$$

$$\nu(t, u, v, y, z) = q(t, u, v, y, z) \cdot \tau(t, u, v, y, z). \quad (23)$$

This is equivalent to the vector \mathbf{v}_τ always being an infinitesimal generator of a symmetry:

$$\mathbf{v}_\tau = \tau\partial_t + f\tau\partial_u + g\tau\partial_v + p\tau\partial_y + q\tau\partial_z. \quad (24)$$

In fact, taking a closer look at this infinitesimal generator we can observe that it corresponds to the symmetry where the flows are exactly the solution curves of the system (10). Thus \mathbf{v}_τ does not move a solution curve into another solution curve, but rather moves *along* the solution curves. (For a more detailed explanation, see Section 3.3.2 of D2.2.9.) This symmetry does not give us any ‘new’ or useful information on the system (10). Therefore let us call the infinitesimal generator \mathbf{v}_τ (for arbitrary $\tau(t, u, v, y, z)$) a *trivial* infinitesimal generator.

The infinitesimal generators form a Lie algebra: i.e. if \mathbf{v}_1 and \mathbf{v}_2 are infinitesimal generators of some symmetries, then $\mathbf{v}_1 + \mathbf{v}_2$ and $[\mathbf{v}_1, \mathbf{v}_2]$ are infinitesimal generators, as well. We define an equivalence relation on the Lie algebra of infinitesimal generators. We call two infinitesimal generators equivalent, if their difference is a trivial infinitesimal generator, i.e. \mathbf{v}_1 and \mathbf{v}_2 are equivalent (denoted by $\mathbf{v}_1 \sim \mathbf{v}_2$) if and only if $\mathbf{v}_1 - \mathbf{v}_2 = \mathbf{v}_\tau$ for some $\tau(t, u, v, y, z)$. It is easy to see that this relation is indeed an equivalence relation, which captures the nontrivial symmetries.

If $\mathbf{v} = \tau\partial_t + \phi\partial_u + \psi\partial_v + \mu\partial_y + \nu\partial_z$ is an infinitesimal generator, then $\mathbf{v} \sim \mathbf{v} - \mathbf{v}_\tau = (\phi - f\tau)\partial_u + (\psi - g\tau)\partial_v + (\mu - p\tau)\partial_y + (\nu - q\tau)\partial_z$. Thus every infinitesimal generator is equivalent to one with coefficient $\tau = 0$. Thus without loss of generality we can assume that $\tau = 0$. Then the symmetry conditions of the system (10) are

$$-f_u\phi - f_v\psi - f_y\mu - f_z\nu + S(\phi) = 0, \quad (25)$$

$$-g_u\phi - g_v\psi - g_y\mu - g_z\nu + S(\psi) = 0, \quad (26)$$

$$-p_u\phi - p_v\psi - p_y\mu - p_z\nu + S(\mu) = 0, \quad (27)$$

$$-q_u\phi - q_v\psi - q_y\mu - q_z\nu + S(\nu) = 0, \quad (28)$$

where S is defined by (15).

E.2 The general p53 model

The detailed description of the p53 model we use here can be found in the OPAALS Deliverable D1.3, Chapter 3. We will use the dimensionless model described in Section 3.3.2 of that report. This system has four dependent variables: Pi (p53), M (mdm2), C (p53-mdm2), Pa (p53*), and each of them depends on the time t . The equations are the following:

$$Pi_t = 1 + \beta_a Pa - (\beta_p + s(t)) Pi - \alpha_c Pi M + \beta_c C, \quad (29)$$

$$M_t = \alpha_{m0} + \frac{\alpha'_{m1} Pi + \alpha'_{m2} Pa}{Pi + Pa + \kappa_m} + (1 + \beta_c) C - (\beta_m + \alpha_c Pi) M, \quad (30)$$

$$C_t = \alpha_c Pi M - (1 + \beta_c) C, \quad (31)$$

$$Pa_t = s(t) Pi - (\beta_a + \beta_p) Pa. \quad (32)$$

Here, all greek letter parameters are constants. Moreover, $s(t)$ represents the radiation stimulus, which induces the creation of Pa -type p53 molecules from Pi -type p53 molecules. The function $s(t)$ can have three different interesting forms depending on three situations:

1. When there is no stimulus at all, i.e. $s(t) = 0$. This can be considered as initial situation, when $Pa = 0$, and Pa will stay 0, as $Pa_t = 0$. This extremely simplifies the system by basically eliminating the variable Pa and the function $s(t)$.
2. When it is kept at a constant level, i.e. $s(t) = \alpha_a$ is constant.
3. When the stimulation is a discrete pulse ‘insult’ at time zero, which leads to $s(t) = \alpha_a e^{-\gamma t}$, with some parameters α_a and γ .

There is another simplification we do, which corresponds to the saturation of the system, i.e. to the expression

$$\frac{\alpha'_{m1}Pi + \alpha'_{m2}Pa}{Pi + Pa + \kappa_m}. \quad (33)$$

As in most situations κ_m is much bigger than Pi and Pa , we simply neglect the latter two in the denominator and replace (33) by

$$\alpha_{m1}Pi + \alpha_{m2}Pa, \quad (34)$$

where $\alpha_{m1} = \alpha'_{m1}/\kappa_m$ and $\alpha_{m2} = \alpha'_{m2}/\kappa_m$. Thus we consider the following system:

$$Pi_t = 1 + \beta_a Pa - (\beta_p + s(t)) Pi - \alpha_c Pi M + \beta_c C, \quad (35)$$

$$M_t = \alpha_{m0} + \alpha_{m1}Pi + \alpha_{m2}Pa + (1 + \beta_c) C - (\beta_m + \alpha_c Pi) M, \quad (36)$$

$$C_t = \alpha_c Pi M - (1 + \beta_c) C, \quad (37)$$

$$Pa_t = s(t) Pi - (\beta_a + \beta_p) Pa. \quad (38)$$

E.3 No stimulus

In this situation we have no radiation stimulus at all, i.e. $s(t) = 0$. Moreover, as it is an initial situation, we have $Pa = 0$, as well. Incorporating it to the equations (35–38) and renaming the variables to u , v and w we obtain

$$u_t = 1 - \beta_p u - \alpha_c uv + \beta_c w, \quad (39)$$

$$v_t = \alpha_{m0} + \alpha_{m1}u - \beta_m v - \alpha_c uv + (1 + \beta_c) w, \quad (40)$$

$$w_t = \alpha_c uv - (1 + \beta_c) w. \quad (41)$$

Now, $\mathbf{v} = \phi(t, u, v, w) \partial_u + \psi(t, u, v, w) \partial_v + \rho(t, u, v, w) \partial_w$ is a symmetry if the functions ϕ , ψ and ρ satisfy the symmetry conditions:

$$(\beta_p + \alpha_c v) \phi + \alpha_c u \psi - \beta_c \rho + S(\phi) = 0, \quad (42)$$

$$(-\alpha_{m1} + \alpha_c v) \phi + (\beta_m + \alpha_c u) \psi - (1 + \beta_c) \rho + S(\psi) = 0, \quad (43)$$

$$-\alpha_c v \phi - \alpha_c u \psi + (1 + \beta_c) \rho + S(\rho) = 0, \quad (44)$$

where the operator S is defined by

$$S(\zeta) = \zeta_t + (1 - \beta_p u - \alpha_c uv + \beta_c w) \zeta_u + (\alpha_{m0} + \alpha_{m1}u - \beta_m v - \alpha_c uv + (1 + \beta_c) w) \zeta_v + (\alpha_c uv - (1 + \beta_c) w) \zeta_w. \quad (45)$$

Adding (44) to equations (42) and (43) and using the additivity of S we obtain

$$\beta_p \phi + \rho + S(\phi + \rho) = 0, \quad (46)$$

$$-\alpha_{m1} \phi + \beta_m \psi + S(\psi + \rho) = 0. \quad (47)$$

Now, if the parameters are $\beta_p = 1 = \alpha_{m1} = \beta_m$, then these equations are further simplified to

$$\phi + \rho + S(\phi + \rho) = 0, \quad (48)$$

$$-\phi + \psi + S(\psi + \rho) = 0. \quad (49)$$

From (48) we immediately obtain that

$$\phi = -\rho + \xi_1, \quad (50)$$

where ξ_1 is a solution to $\xi_1 + S(\xi_1) = 0$. Substituting it into (48) we obtain that

$$\psi = -\rho + \xi_2, \quad (51)$$

where ξ_1 and ξ_2 satisfy

$$\xi_1 + S(\xi_1) = 0, \quad (52)$$

$$-\xi_1 + \xi_2 + S(\xi_2) = 0. \quad (53)$$

To find the general solutions to equations (52) and (53) can be really hard, but we were able to find solutions by chance:

$$\xi_1 = c_1 \cdot e^{-t}, \quad (54)$$

$$\xi_2 = c_1 \cdot t \cdot e^{-t} + c_2 \cdot e^{-t}. \quad (55)$$

Inspired by the fact that we can find nice nontrivial examples for the functions $\xi_1 = \phi + \rho$ and $\xi_2 = \psi + \rho$, we introduce the following coordinate change:

$$y(t) = u(t) + w(t), \quad (56)$$

$$z(t) = v(t) + w(t). \quad (57)$$

With these new coordinates we have $u = y - w$, $v = z - w$ and equations (42–44) become

$$y_t - w_t = 1 - y + w - \alpha_c(y - w) \cdot (z - w) + \beta_c w, \quad (58)$$

$$z_t - w_t = \alpha_{m0} + y - w - (z - w) - \alpha_c(y - w) \cdot (z - w) + (1 + \beta_c)w, \quad (59)$$

$$w_t = \alpha_c(y - w) \cdot (z - w) - (1 + \beta_c)w, \quad (60)$$

which is equivalent to the system

$$y_t = 1 - y, \quad (61)$$

$$z_t = \alpha_{m0} + y - z, \quad (62)$$

$$w_t = \alpha_c w^2 - (1 + \beta_c + \alpha_c y + \alpha_c z)w + \alpha_c yz. \quad (63)$$

Here, the solution to (61) is almost the same function as ξ_1 :

$$y(t) = 1 + c_1 \cdot e^{-t}. \quad (64)$$

Substituting this solution into (62) we obtain a solution for $z(t)$ similar to ξ_2 :

$$z(t) = 1 + \alpha_{m0} + c_1 \cdot t \cdot e^{-t} + c_2 \cdot e^{-t}. \quad (65)$$

Thus we have reduced our original system (42–44) to the Riccati equation

$$\boxed{w_t = \alpha_c w^2 - (1 + \beta_c + \alpha_c (2 + \alpha_{m0} + (c_1 + c_2) \cdot e^{-t} + c_1 \cdot t \cdot e^{-t}))w + \alpha_c \cdot (1 + c_1 \cdot e^{-t}) \cdot (1 + \alpha_{m0} + c_1 \cdot t \cdot e^{-t} + c_2 \cdot e^{-t})}. \quad (66)}$$

Had we found a solution to equation (66), we could express $u(t)$ and $v(t)$ in the following way:

$$u(t) = 1 + c_1 \cdot e^{-t} - w(t), \quad (67)$$

$$v(t) = 1 + \alpha_{m0} + c_1 \cdot t \cdot e^{-t} + c_2 \cdot e^{-t} - w(t). \quad (68)$$

The constants c_1 and c_2 are determined by the initial conditions:

$$c_1 = u(0) + w(0) - 1, \quad (69)$$

$$c_2 = v(0) + w(0) - 1 - \alpha_{m0}. \quad (70)$$

Unfortunately we cannot solve equation (66) in general, unless we can find at least one solution. Nevertheless, we can try to solve it numerically. Figure F shows the solution curve for the following choice of parameters (see OPAALS Deliverable D1.3 [DBLML09]):

$$u(0) = 3.5 \quad v(0) = 0.008 \quad w(0) = 0.993 \quad (71)$$

$$\alpha_{m0} = 0.00005 \quad \alpha_c = 35 \quad \beta_c = 0.005. \quad (72)$$

One can read from the figure that w increases in time until it reaches a maximum, then it decreases, which is consistent with its decaying factor.

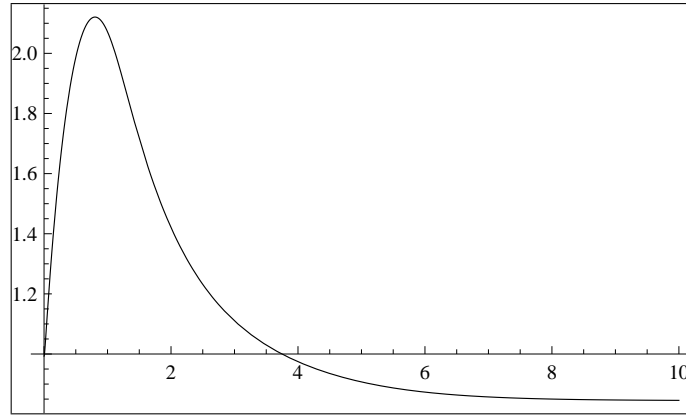


Figure F: Numerical solution to (66)

E.4 No stimulus with initial Pa value

Let us assume now that there is no radiation, i.e. $s(t) = 0$, but unlike before there exists some active p53 initially, i.e. $Pa(0) = a_0 > 0$. Incorporating it into Equations (35–38) and renaming the variables to u , v , w and x we obtain

$$u_t = 1 - \beta_p u + \beta_a x - \alpha_c uv + \beta_c w, \quad (73)$$

$$v_t = \alpha_{m0} + \alpha_{m1} u + \alpha_{m2} x - \beta_m v - \alpha_c uv + (1 + \beta_c) w, \quad (74)$$

$$w_t = \alpha_c uv - (1 + \beta_c) w, \quad (75)$$

$$x_t = -(\beta_a + \beta_p) x. \quad (76)$$

Again, we can introduce the same coordinate change (56) and (57) inspired by the examination of the symmetry conditions of the system:

$$y(t) = u(t) + w(t), \quad (77)$$

$$z(t) = v(t) + w(t). \quad (78)$$

Using this coordinate change we obtain

$$x_t = -(\beta_a + \beta_p) x, \quad (79)$$

$$y_t = 1 + \beta_a x - \beta_p y + (\beta_p - 1) w, \quad (80)$$

$$z_t = \alpha_{m0} + \alpha_{m2} x + \alpha_{m1} y - \beta_m z + (\beta_m - \alpha_{m1}) w, \quad (81)$$

$$w_t = \alpha_c w^2 - (1 + \beta_c + \alpha_c y + \alpha_c z) w + \alpha_c y z. \quad (82)$$

Again, we observe that if $\beta_p = 1$ and $\alpha_{m1} = \beta_m$, then the equations (79–81) can be solved after each other. In particular the solution for (79) is

$$x(t) = c_1 \cdot e^{-(1+\beta_a)t}, \quad (83)$$

for an arbitrary constant c_1 . Substituting this result into (80) we obtain a solution:

$$y(t) = 1 - x(t) + c_2 \cdot e^{-t} \quad (84)$$

$$= 1 - c_1 \cdot e^{-(1+\beta_a)t} + c_2 \cdot e^{-t}, \quad (85)$$

for an arbitrary constant c_2 . Substituting this result into (81) we obtain a solution for $z(t)$. If $\beta_m \neq 1$ and $\beta_m \neq 1 + \beta_a$, then

$$z(t) = \frac{\alpha_{m0}}{\beta_m} + \frac{1}{(1 - \beta_m)} + \frac{\alpha_{m2}\beta_m - \alpha_{m2} - \beta_a\beta_m}{(\beta_m - 1 - \beta_a)(\beta_m - 1)} \cdot x(t) + \frac{\beta_m}{\beta_m - 1} \cdot y(t) + c_3 \cdot e^{-\beta_m t} \quad (86)$$

$$= 1 + \frac{\alpha_{m0}}{\beta_m} + \frac{\beta_m - \alpha_{m2}}{1 + \beta_a - \beta_m} \cdot c_1 \cdot e^{-(1+\beta_a)t} + \frac{\beta_m}{\beta_m - 1} \cdot c_2 \cdot e^{-t} + c_3 \cdot e^{-\beta_m t}. \quad (87)$$

for an arbitrary constant c_3 . If $\beta_m = 1 + \beta_a$, then

$$z(t) = 1 + \frac{\alpha_{m0}}{1 + \beta_a} + (\alpha_{m2} - 1 - \beta_a) \cdot c_1 \cdot t \cdot e^{-(1+\beta_a)t} + \frac{1 + \beta_a}{\beta_a} \cdot c_2 \cdot e^{-t} + c_3 \cdot e^{-(1+\beta_a)t}, \quad (88)$$

for an arbitrary constant c_3 . If $\beta_m = 1$, then

$$z(t) = 1 + \alpha_{m0} + \frac{1 - \alpha_{m2}}{\beta_a} \cdot c_1 \cdot e^{-(1+\beta_a)t} + c_2 \cdot t \cdot e^{-t} + c_3 \cdot e^{-t}, \quad (89)$$

for an arbitrary constant c_3 . If $\beta_m = 1$ and $\beta_a = 0$, then

$$z(t) = 1 + \alpha_{m0} + (1 - \alpha_{m2}) \cdot c_1 \cdot e^{-2t} + c_2 \cdot t \cdot e^{-t} + c_3 \cdot e^{-t}, \quad (90)$$

for an arbitrary constant c_3 .

Substituting these solutions into (82) we reduce the original system (42–44) to a rather complicated Riccati equation. Had we found a solution to equation (66), we could express $u(t)$ and $v(t)$ in the following way:

$$u(t) = y(t) - w(t), \quad (91)$$

$$v(t) = z(t) - w(t). \quad (92)$$

The constants c_1, c_2, c_3 are determined by the initial conditions $x(0), u(0), v(0), w(0)$:

$$c_1 = x(0), \quad (93)$$

$$c_2 = x(0) + u(0) + w(0) - 1, \quad (94)$$

and if $\beta_m \neq 1$ and $\beta_m \neq 1 + \beta_a$, then

$$c_3 = \left(\frac{\beta_m}{\beta_m - 1} + \frac{\beta_m - \alpha_{m2}}{\beta_m - 1 - \beta_a} \right) \cdot x(0) + \frac{\beta_m}{1 - \beta_m} \cdot u(0) + v(0) + \frac{1}{1 - \beta_m} w(0) \\ + \frac{\beta_m}{\beta_m - 1} + \frac{\beta_a + 1 - \beta_m}{\beta_m - 1 - \beta_a} \cdot \left(1 + \frac{\alpha_{m0}}{\beta_m} \right). \quad (95)$$

If $\beta_m = 1 + \beta_a$, then

$$c_3 = - \left(1 + \frac{1}{\beta_a}\right) \cdot x(0) - \left(1 + \frac{1}{\beta_a}\right) \cdot u(0) + v(0) - \frac{1}{\beta_a} \cdot w(0) + \frac{1}{\beta_a} - \frac{\alpha_{m0}}{1 + \beta_a}. \quad (96)$$

If $\beta_m = 1$, then

$$c_3 = \frac{\alpha_{m2} - 1}{\beta_a} \cdot x(0) + v(0) + w(0) - 1 - \alpha_{m0}. \quad (97)$$

If $\beta_m = 1$ and $\beta_a = 0$, then

$$c_3 = (\alpha_{m2} - 1) \cdot x(0) + v(0) + w(0) - 1 - \alpha_{m0}, \quad (98)$$

Figure G shows the solution curve for the following choice of parameters (see OPAALS Deliverable D1.3 [DBLML09]):

$$x(0) = 0 \quad u(0) = 3.5 \quad v(0) = 0.008 \quad (99)$$

$$w(0) = 0.993 \quad \alpha_{m0} = 0.00005 \quad \alpha_c = 35 \quad (100)$$

$$\beta_c = 0.005 \quad \beta_m = 1 \quad \beta_a = 0.5. \quad (101)$$

It looks quite similar to the one shown in Figure F, i.e. an overdamped system.

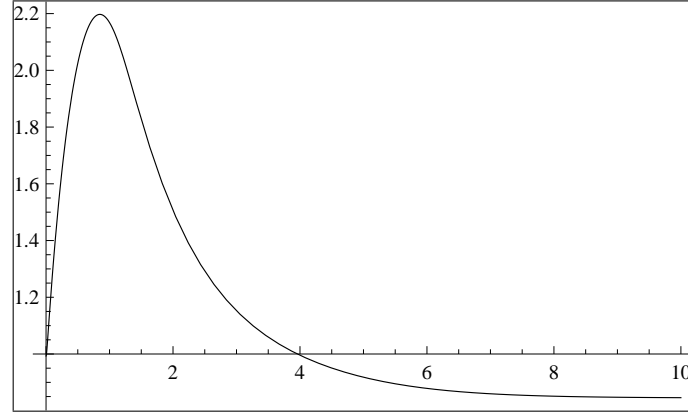


Figure G: Numerical solution to (82)

E.5 Conclusion

It is interesting that by making use of the transformation

$$w(t) = r'(t)/r(t) \quad (102)$$

the Riccati equations (66) and (82) can be transformed into linear ODEs in the new variable $r(t)$. However, because the coefficients of these ODEs are complicated functions of time, neither appears to be integrable. Thus for this example we conclude that the Lie symmetry analysis has not been particularly useful, beyond helping us see the possibility to reduce the original problem to a Riccati equation.

Rather than an indication that there are no dynamical symmetries hiding in this regulatory pathway, this result is more likely a consequence of the simplicity of the p53-mdm2 ODE system chosen to represent it. More precisely, from the point of view of a mathematical model that is meant to capture the most important aspects of the physical phenomenon this model might be too simple. On the other hand, as a mathematical problem it is already rather complicated and practically impossible to solve analytically. So we need to look for a better model that is more expressive physically but simpler mathematically, which is not easy to do. More variables are in general used as part of this pathway, so that the next logical step is to enlarge the system to 5 or 6 equations. The present analysis has already helped us see quite a few features of the problem, and therefore provides a starting point upon which to build a more in-depth investigation.